TECHNISCHE
UNIVERSITÄT
DARMSTADT

User-centric Video in the Future Internet: QoE in Participatory Video Generation and Distribution

Dem Fachbereich Informatik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
vorgelegte Dissertation

von

Denny Stohr, M.Sc.

Geboren am 19. Juli 1988 in Mainz, Deutschland

Erstgutachter: Prof. Dr.-Ing. Ralf Steinmetz
Zweitgutachter: Prof. Dr.-Ing. Wolfgang Effelsberg

Darmstadt, 2018

Stohr, Denny : User-centric Video in the Future Internet: QoE in Participatory Video Generation and Distribution, Technische Universität Darmstadt
Jahr der Veröffentlichung der Dissertation auf TUprints: 2018

Tag der mündlichen Prüfung: 16.07.2018

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-76162
URL: https://tuprints.ulb.tu-darmstadt.de/id/eprint/7616

Abstract

Today, more than 73 percent of all transmitted data on the Internet is video traffic, making it the central network application which is used by billions of users globally; with new service offerings, improving content quality and, an increasing number of customers, also, new challenges arise in this domain. For example, streamed video is viewed and shared more than ever on mobile devices, bandwidth requirements rise to support standards with superior qualities like 4K and HDR, and worldwide service offerings come with diverse network environments to handle.

Driven by these challenges, this dissertation presents research with the central goal to measurably improve users' Quality of Experience in current and future video applications on the Internet. We present findings in integral parts of video streaming applications, comprising adaptive live mobile broadcasting and video on demand use cases within three integrative research areas.

In our first contribution, we initially present results of a measurement study on live mobile video broadcasting services that show the video upload quality to be particularly impaired when mobile connections are used. For the automatic composition of live video, the quality of such mobile broadcasts are a prerequisite for achieving a high user satisfaction by switching between the best available content from multiple sources. However, the current approach to upload all available live user-generated video streams for mobile video composition leads to a high overhead on mobile devices. Our work presents a new method based on device context measurements that allows to drastically improve efficiency in such automatic video composition systems by identifying the relevant quality indicators on the device based on derived sensor and network measurements. We achieve an improved Quality of Experience with our proposed context-based stream selection method as verified in a field test and a crowd-sourced user study.

Next, in the context of the distribution of video on demand content using Dynamic Adaptive Streaming over HTTP (DASH), we show that strong potential lies in investigating the cross-layer configuration space of video streaming systems, given the wide range of interdependent system aspects, environments, and service requirements as opposed to state-of-the-art research that focuses on single system aspects such as adaptation algorithms. By generating a broad set of experiments, i. e., covering a wide spectrum of cross-layer DASH video streaming system configuration parameters, we identify such performance aspects related to, e. g., the TCP congestion control, adaptation algorithms, and DASH players within heterogeneous network environments. We show that a subset of concrete configurations can improve DASH user experience in video on demand applications, and further motivate transitions of such DASH mechanisms based on learned sweet spot configurations.

Last, we envision that in the long term, more fundamental changes to the underlying network infrastructure of the Internet need to be considered for addressing the demands of developing video streaming systems by investigation of adaptive video distribution in Named Data Networks (NDNs). First, we show that the naïve application of established concepts in DASH adaptation algorithms, that use buffer or segment throughput measurements as input, lead to unfavorable results given substantial differences in the network behavior of NDN. Our proposed concept for adaptation algorithms in NDNs is based on an improved network throughput measurement method and is shown to reduce stalling and increase streaming bitrates as compared to approaches used in current DASH adaptation algorithms.

Overall, this dissertation provides the following contributions: *i)* first, a detailed emulation-based analysis and comparison of today's DASH system implementations and algorithms, *ii)* novel concepts to enable efficient live mobile video composition, *iii)* and last, significant improvements in the performance for adaptive video streaming systems with the emerging NDN paradigm.

Kurzfassung

Heutzutage macht das Streamen von Videos mehr als 73 Prozent aller übertragenen Daten im Internet aus. Das macht es zu der zentralen Anwendung in den existierenden Netzwerken, die weltweit von Milliarden von Nutzern genutzt wird. Mit neuen Serviceangeboten, steigender Content-Qualität und einer ebenso steigenden Anzahl von Kunden ergeben sich auch neue Herausforderungen in diesem Bereich. Zum Beispiel werden Videos mehr als je zuvor auf mobilen Geräten angesehen und geteilt, Bandbreitenanforderungen steigen durch Standards mit höheren Qualitäten wie 4K und HDR und zuletzt geht das weltweite Anbieten von Videodiensten auch mit unterschiedlichen Netzwerkumgebungen einher, welche von Dienstanbietern bewältigt werden müssen.

Vor diesem Hintergrund präsentiert die vorliegende Dissertation Forschung mit dem zentralen Ziel, die wahrgenommene Qualität, also Quality of Experience (QoE), von Nutzern in aktuellen und zukünftigen Videoanwendungen im Internet messbar zu verbessern. Wir präsentieren Ergebnisse in integralen Teilen von Video-Streaming-Anwendungen, die adaptive Live-Broadcasting und Video-on-Demand (VoD) Anwendungsfälle in drei integrativen Forschungsbereichen umfassen.

In unserem ersten Beitrag präsentieren wir zunächst eine Messungsstudie zu Live-Video-Broadcasting-Diensten und stellen fest, dass Video-Uploads eine niedrige Gesamt-Videoqualität aufweisen, insbesondere wenn sie von mobilen Verbindungen übertragen werden. Für die automatische Zusammenstellung von Live-Videos ist die Qualität solcher mobilen Übertragungen eine Voraussetzung, um eine hohe Benutzerzufriedenheit zu erreichen, indem zwischen den besten verfügbaren Inhalten aus mehreren Quellen umgeschaltet wird. Der derzeit verwendete Ansatz zum Hochladen aller verfügbaren Live-User-Generated Videos für Mobile Video Kompositen führt zu einer sehr hohen Datennutzung auf mobilen Geräten. Unsere Arbeit stellt eine neue Methode vor, die auf Messungen von Gerätekontexten beruht und es ermöglicht, die Effizienz in solchen automatischen Kompositionssystemen drastisch zu verbessern, indem die relevanten Qualitätsindikatoren auf dem Gerät basierend auf abgeleiteten Sensor- und Netzwerkmessungen identifiziert werden. Wir erreichen eine verbesserte QoE mit unserer vorgeschlagenen kontextbasierten Stream-Auswahlmethode, die in einem Feldtest und einer Crowdsourcing-Benutzerstudie verifiziert wurde.

Als Nächstes zeigen wir, dass bei der Verteilung von VoD-Inhalten mit dynamischem adaptivem Streaming über HTTP (DASH) durch die Untersuchung des Cross-Layer-Konfigurationsraums des Videos Streaming Systems (VSS) weitere Serviceverbesserungen möglich sind. Im Gegensatz zur Fokussierung auf einzelne Systemaspekte wie Adaptierungsalgorithmen lassen sich hier, angesichts der großen Bandbreite an voneinander abhängigen Systemaspekten, Umgebungen

und Serviceanforderungen Abhängigkeiten zwischen konkreten Konfigurationen finden, die eine höhere QoE für Kunden erzielen können. Durch das Generieren einer breiten Reihe von Experimenten, d.e., die ein breites Spektrum von schichtübergreifenden DASH-VSS-Konfigurationsparametern abdeckt, identifizieren wir solche Leistungsaspekte, die z. B. die TCP Congestion Control und DASH Player in heterogenen Netzwerkumgebungen berücksichtigen. Wir zeigen, dass eine Teilmenge konkreter Konfigurationen die DASH QoE in VoD-Anwendungen verbessern und Übergänge solcher DASH-Mechanismen basierend auf erlernten Sweet-Spot-Konfigurationen weiter motivieren kann.

Schließlich ist unsere Meinung, dass auf lange Sicht grundlegendere Änderungen an der zugrunde liegenden Netzwerkinfrastruktur des Internets in Betracht gezogen werden müssen, um die Anforderungen der Entwicklung von VSS zu bewältigen. Hier sehen wir insbesondere die Untersuchung und Anwendung der adaptiven Videoverteilung in Named Data Networking (NDNs) als geeignete Methode. Zunächst zeigen wir, dass die naive Anwendung etablierter Konzepte in DASH, die Puffer- oder Segmentdurchsatzmessungen als Input verwenden, zu ungünstigen Ergebnissen führen, wenn das Netzwerkverhalten von NDN stark variiert. Unser vorgeschlagenes Konzept für NDNs basiert auf einer verbesserten Netzwerk-Durchsatz-Messmethode und reduziert nachweislich das Nachladen von Videoinhalten und erhöht die Streaming-Bitraten im Vergleich zu aktuellen Ansätzen in DASH.

Insgesamt liefert diese Dissertation die erste detaillierte emulationsbasierte Analyse und einen Vergleich der heutigen adaptiven VSS-Implementierungen und -Algorithmen, führt neue Konzepte ein, um eine effiziente Live-Mobile Video Kompositen zu ermöglichen und verbessert signifikant die Leistung für adaptives Video-Streaming mit dem aufkommenden NDN-Paradigma.

Acknowledgments

I want to extend my sincere gratitude to the people supporting me with this dissertation (in order of our first encounter):

Wolfgang Effelsberg: For guiding me in my entire academic career, from being the lecturer of *Praktische Informatik I* in my first semester as a student at University of Mannheim, advising me in my Master's Thesis, introducing me to the opportunity to peruse a Ph.D., to being my adviser and mentor during my Ph.D. studies at TU Darmstadt. You have taught me not only academic lessons, but showed me the virtues of modesty, patience, and facing challenges with optimism.

Andreas Mauthe: For spiking my excitement for academic collaboration in my student visit to Lancaster University and the guidance as my adviser of subproject C3 in MAKI.

Ralf Steinmetz: For giving me the opportunity and trust to peruse a Ph.D. at KOM and being my primary adviser during this time.

Stefan Wilk: As my friend, colleague, and co-author. Thank you for guidance and support during our time at KOM/DMS and the calm atmosphere you provided to our shared office during those years.

Alexander Frömmgen: For the memorable last year of our Ph.D.'s, as friends, colleges, and co-authors.

Amr Rizk: For his patience and academic mindset as my supervisor and co-author.

Of course, there were many other amazing people I was fortunate to meet during this time, in particular, my friends and colleagues at KOM; student assistants that helped me with much of the technical challenges of conducting research, in particular Iva Toteva and Timo Kalle; and many motivated students I worked with in Theses and Seminars.

This work was also supported by the Collaborative Research Centre "MAKI"and the Profile Area Internet and Digitization which provided me with excellent resources, e. g., to meet amazing international researchers to share and discuss ideas on the many conferences I was fortunate to visit.

Last, none of this would have been possible without the love, support, and care of my parents, family, and wife Gergana. Thank you all.

Denny Stohr
Darmstadt
June 2018

Contents

1 Introduction 1
1.1 Research Challenges . . . 3
1.2 Research Goals . . . 4
1.3 Thesis Outline . . . 4
1.4 Previously Published Material . . . 5

2 Fundamentals and Definitions 6
2.1 Networking Concepts . . . 6
2.1.1 Networking Protocols . . . 6
2.1.2 Software-Defined Radio . . . 8
2.1.3 Information Centric Networking . . . 9
2.2 Over-the-Top Video Delivery . . . 11
2.2.1 Video Codecs . . . 11
2.2.2 Scalable Video Coding . . . 13
2.2.3 Emerging Codecs . . . 13
2.2.4 Push-based Real-Time Streaming . . . 13
2.2.5 Adaptive Streaming over HTTP . . . 14
2.3 User-Generated Video . . . 17
2.4 Performance Evaluation . . . 18
2.4.1 Quality of Service . . . 19
2.4.2 Quality of Experience . . . 22

3 Related Work 25
3.1 Mobile Video Upload, Distribution, and Composition . . . 25
3.1.1 Mobile Video Broadcasting Services . . . 25
3.1.2 Live Video Streaming Services . . . 26
3.1.3 Video Composition . . . 27
3.1.4 Discussion . . . 29
3.2 Network Interdependencies in OTT Video Streaming . . . 29
3.2.1 Overview of DASH Adaptation Algorithms . . . 29
3.2.2 Network Interdependency Effects in DASH . . . 32
3.2.3 Discussion . . . 33
3.3 Video Streaming in Novel Network Architectures . . . 34
3.3.1 NDN-based DASH . . . 34
3.3.2 SDR-based Adaptive Video Streaming . . . 35
3.3.3 New Transport Protocols for Dynamic Adaptive Streaming 35
3.3.4 Discussion . . . 36

4 Mobile User-Generated Video Composition and Distribution 37
4.1 Analysis of YouNow . . . 38
4.2 YouNow vs. other Mobile Video Broadcasting Services . . . 38
4.2.1 YouNow Architecture . . . 39
4.2.2 Broadcasting and Viewing Workload . . . 40
4.2.3 Session Duration and Online Time of Broadcasters . . . 43
4.2.4 Broadcaster Popularity . . . 44
4.2.5 Device Usage . . . 44
4.2.6 Broadcasting Quality . . . 45
4.3 Context-Based Mobile Video Composition Support . . . 47
4.3.1 Context-Based Metrics and Data Collection . . . 48
4.3.2 Protocol and Architecture . . . 50
4.3.3 The Video Recording Application: LiViU . . . 51
4.3.4 Composition . . . 53
4.4 Evaluation . . . 54
4.4.1 Evaluation Setup . . . 54
4.4.2 Crowd-sourced User Study on Quality of Experience . . . 57
4.4.3 Real-time Composition Capability . . . 59
4.5 Conclusions . . . 60

5 Network Interdependencies in HTTP Adaptive Streaming 61
5.1 Extensive Emulations for DASH . . . 62
5.1.1 Evaluation Framework Architecture . . . 62
5.1.2 DASH Emulation Environment . . . 63
5.2 Cross Layer Environment and Configurations . . . 65
5.2.1 Network Conditions . . . 65
5.2.2 Transport Protocol . . . 65
5.2.3 Adaptation Algorithms . . . 65
5.2.4 DASH Players . . . 66
5.2.5 Video Content . . . 68
5.2.6 QoS and QoE Metrics . . . 68
5.3 Evaluation Concept for DASH . . . 68
5.3.1 The Case for an Emulation-based Evaluation of DASH . . . 69
5.3.2 DASH QoS: Cross-Layer Performance Characteristics . . . 71
5.3.3 QoE-Centric Analysis and Design Trade-Offs . . . 79
5.3.4 On Improving QoE Trough Transitions . . . 83
5.4 Conclusions . . . 85

6 Video Streaming in Future Networks 87
6.1 Motivation for NDN-based DASH Delivery . . . 88
6.2 Emulative Evaluation of DASH in Named Data Networks . . . 89
6.2.1 Network Environment . . . 89
6.2.2 NDN Mechanisms . . . 90
6.2.3 DASH Mechanisms . . . 91
6.3 Challenges for DASH over NDN . . . 91
6.4 Approach for Improved DASH Adaptation in NDNs . . . 93
6.4.1 Chunk-based Measurements . . . 93

6.4.2 Extended MPD 95
6.4.3 Adaptation Hysteria Reduction 95
6.5 Evaluation of Chunk-Aware DASH Adaptation in NDN 96
6.6 Practical Feasibility of NDN-based DASH Streaming 99
6.7 Conclusions 100

7 Conclusions and Outlook **102**
7.1 Thesis Summary 102
7.2 Contributions 103
7.3 Outlook 104
7.3.1 Extending DASH Research using other Players 104
7.3.2 Applying DASH Reconfigurations in Practice 104
7.3.3 NDNs for Mobile Video Composition 105
7.3.4 Merging Context and Content in Mobile Video Composition 105

Bibliography **xi**

Author's Publications **xxvii**

Curriculum Vitæ **xxxiii**

1 Introduction

In recent years and decades, one might observe that we have been living through a time of an unprecedented rate of technological advancements; for instance, the advent of smart-phones, often anecdotally marked by Apple's introduction of the first iPhone in 2007[a,b], has been shaping many aspects of our day-to-day lives. Soon almost three billion people will be using such devices, many spending several hours of their day interacting with it.[c,d] As a result, information is now always accessible and mere telephony can now be enhanced by video and a high degree of interaction; hence we live in a time of rapid multimedia communication where rich content can be shared instantly between individuals and groups of users worldwide.

One might argue that one property, in particular, has been the primary driver of this development: while many forms of access exist, from wired stationary networks to WiFi and the wide-spread coverage of mobile Internet, they all allow access to the same medium to share and access information. Thus, it is claimed that the single most disruptive technology of this century has been the Internet [MR11]. In the wake of this success, Internet-driven technologies and services transformed many long-standing norms in society, from how information is accessed, the way people communicate, and content is produced and shared. With this, one area that has undergone a particular significant transformation is video.

Whereas in the past most content was produced by television stations and production studios, broadcasted via satellites or other dedicated channels, and consumed linearly, each of those steps is in the process of becoming obsolete. Today, content is produced by a variety of commercial and private sources and caters to an increasingly individual desire of users. To name just some examples in today's wide-ranging landscape *i)* users may choose amateur recordings or live-streams uploaded directly from mobile devices, (e. g., Facebook Live or YouNow); *ii)* they can select from ever growing range of highly professionalized (and monetized) channels on YouTube with millions of regular viewers;[1] *iii)* view large-budget professional productions by streaming providers such as Netflix *iv)* and also access content from classical studios and television stations that transform their offers to an increasingly mobile-centered audience. The technological advancements that allowed for all of these very different use cases can be subsumed to a single term: *the Internet*.

[1] *As of March 2018, the channel with most subscribers of 61 000 000, is PewDiePie on YouTube.*

[a] http://web.archive.org/web/20070110052128/http://www.apple.com:80/

[b] http://content.time.com/time/covers/0,16641,20071112,00.html

[c] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[d] https://www.statista.com/statistics/781692/worldwide-daily-time-spent-on-smartphone/

With its decentralized architecture and governance, content can be shared between any two connected end hosts, to a large degree with no direct relation between the producer and the transmission medium. Thus, for this transmission to be measurably successful and subjectively satisfying for users—a concept summarized by the term Quality of Experience (QoE)—many different parties, having different interests[2] and technologies have to work seamlessly together. This form of video distribution—that is *Over-The-Top (OTT)* of the Internet—accounts for the largest share of worldwide network data traffic, as measured by Cisco [Cis17]; their predictions indicate an increase of this traffic share to 81 % of all data transmissions in the Internet along with a continuing rise of traffic volume. Hence, the total estimate of traffic taken up by video transmission services over the Internet is estimated to become 3.3 ZB in 2021. Putting this number into perspective, even when making the assumption that all content is transmitted in high definition quality[3], this amount of data corresponds to 1 100 000 000 000 hours of video transmission or about 137.5 h per capita in each year worldwide.

[2] *This also relates to the discussion regarding net neutrality.*

[3] *Corresponding to 3 GBph of transmitted data according to Netflix.*[e]

With all its benefits allowing for a dynamic, individualized, content distribution considered, there are some evident drawbacks to Internet-based transmission. Traditional broadcasting technology allowed to reach millions of end devices (i. e., CRT television sets) through delivering key events like the Apollo moon landing.[f,4] Replicating this scale is still very challenging, and costly for today's streaming providers, given the prevalent method of content distribution makes use of unicast transmissions between a Content Delivery Network (CDN)'s node and individual users. Instead of a dedicated and (up to the end user) entirely controlled infrastructure (e. g. sending stations, satellites, and stationary receivers attached to TVs), each consumer requires an own share of capacity in an OTT streaming architecture.[5] Yet, today's customers expect a high QoE even when they are consuming content in a very dynamic fashion; a user nowadays might be watching high-definition content on Smart TVs streamed using Netflix, short-form content during their bus commute with their smartphone on YouTube, or even spontaneously live streaming from her mobile device to millions of other users worldwide on Facebook.live.

[4] *The moon landing event has been watched by an estimated number of 530 000 000 people worldwide. For today's population, this corresponds to approximately 200 Tbps of capacity required for transmitting this event in HD definition to households of four. The highest recorded request volume of Akamai to date is 60 Tbps.*

[5] *While alternative strategies exist, such as P2P streaming and multicast, they are not widely used in OTT streaming. Other managed architectures, such as IPTV, deploy such concepts.*

To make this possible, each of the given examples entails unique challenges to be solved that become more complex with current developments in demand and market competition. With providers increasingly moving into markets where mobile usage is central (i. e., Asia and Africa), addressing challenges related to maintaining a high QoE—even when deployed in more volatile mobile networks—becomes a central aspect of prevailing in this competitive market. Likewise, ensuring higher efficiency and quality when streaming live user-generated content is still an area open for vast improvements along with exploring future potentials based on new networking paradigms that promise higher service quality and efficiency in such services. While acknowledging that this domain is rapidly evolving, driven by a large body of academic and commercial interest, this work aims to propose, design, implement, and evaluate novel concepts dedicated to each of those areas that will provide a tangible improvement for users of such systems in the today and in the future.

[e] https://help.netflix.com/en/node/87

[f] https://www.nasa.gov/mission%5C_pages/apollo/missions/apollo11.html

1.1 Research Challenges

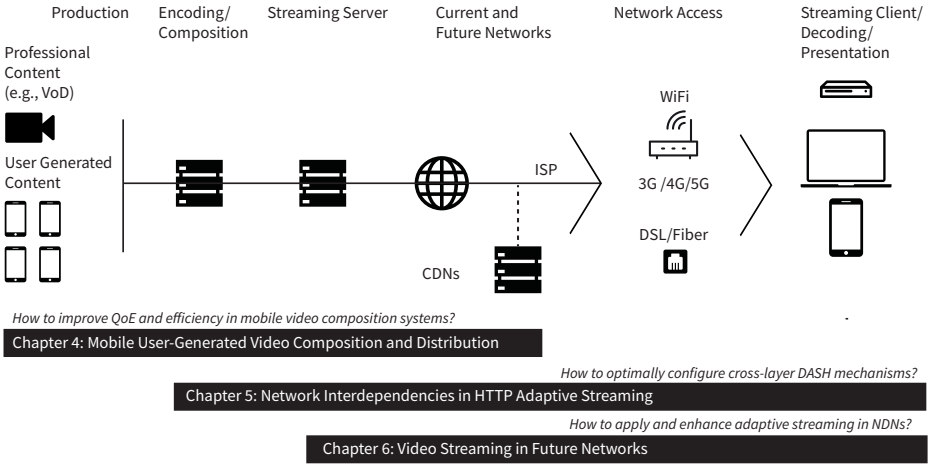


Figure 1: Overview of Internet-based OTT-streaming architecture components used in MBS, DASH and Future Networks related to chapters presenting novel research in these areas in this dissertation.

Satisfying the before-mentioned user demands along with the expected growth in traffic volume, and rising significance of user-centric video streaming translates to a set of challenges that require being investigated from multiple angles, as depicted in Figure 1.

First, more advanced methods on how to measure and improve user satisfaction, in ever more dynamic and varied system architectures, is a core challenge to be addressed. With that, research needs to investigate the dynamic creation of content, as seen in User-Generated Video, and explore methods to understand the complex nature for the efficient distribution of video considering unmanaged (i. e., Over-The-Top) distribution approaches. Last, making such services viable with the increasing demand, in the long term, new networking paradigms may better suit the need of a high and dynamic request volume but require careful investigation for their suitability to adhere to the given service quality level.

To this end, this dissertation seeks to tackle the core questions in each of these areas and thus provide a multi-angled view on the discussed challenges as follows:

Efficiency in the dynamic creation and distribution of user-generated video: With the vast amount of continuously produced User-Generated Video, the selection and distribution of relevant, high-quality content is very challenging, given the constraints on mobile data upload capabilities, short individual recording time of users and very heterogeneous quality of mobile, live User-Generated Content. Thus, the efficient selection, composition, and upload of such content need to be addressed in research.

High user satisfaction for over-the-top video distribution using adaptive streaming over HTTP: For video distribution on the Internet, providers rely on unmanaged networks in the form of OTT service provisioning based on the HTTP Adaptive Streaming (HAS) paradigm.

Given the dynamic nature of such networks, providing consistent QoE is very challenging; the interdependence between network's characteristics and configurations has to be explored to realize a consistent high QoE along with evolving user demands, especially in heterogeneous network conditions.

Integration of future networking concepts for adaptive video streaming: Upcoming network technologies such as Named Data Networking (NDN) promise benefits for Adaptive Video Streaming (AVS); realizing this potential requires to verify their applicability, investigate identified challenges, and demonstrate performance improvements.

1.2 Research Goals

The overarching goal of this dissertation is to *evolve user-centric video applications by identifying potentials in cross-layer configurations and the usage context in adaptive video streaming systems to measurably enhance QoE in current and Future Internet architectures.* We divide our research objective into the following subgoals:

Research Goal 1: Analyze and improve QoE in Mobile Video Composition (MVC) while reducing the currently prevalent high overhead for required live upload of User-Generated Video to minimize costs for users.

Research Goal 2: Identify key influence factors and optimized configurations in adaptive streaming scenarios for heterogeneous networks using OTT service delivery.

Research Goal 3: Apply and enhance current adaptive streaming concepts to the Future Internet architecture NDN to improve the resilience and efficiency of Video Streaming Systems (VSSs).

1.3 Thesis Outline

We begin this dissertation with an introduction to relevant background concepts. Here, we present the relevant areas oriented on the Internet protocol stack, starting with PHY-Layer, Software-Defined Radio (SDR), and NDN networking concepts. We then proceed to a discussion of transport protocols, in particular, TCP Congestion Controls (CCs).

Moving on to the application layer, we present push and pull-based video transport protocols central to this thesis. Here we focus in particular on DASH as this is one of the central protocols researched in this dissertation. The last concepts presented in the background are related to the Quality of Service (QoS) and QoE of systems presented in this thesis as this is crucial for the final performance evaluation for the research conducted in this thesis.

Then we introduce related work corresponding to the three research areas (and Chapters 4 to 6) presented and narrow down the research gaps in accordance with our specified goals that addressed by research in this thesis.

As the first research result in this dissertation (Chapter 4), we analyze existing MBS and proceed to propose and evaluate a novel, context-based approach to identify high QoE sources for live video composition, along with a field-study based evaluation.

The second contribution in Chapter 5 introduces a new concept for a large-scale evolution of DASH scenarios that allows to explore an extended evaluation space and identify *sweet spots* in the configuration of OTT DASH scenarios towards an improved QoE.

The last contribution in Chapter 6 then explores such selected DASH mechanisms in NDN networks and proposes new adaptation concepts related to the particular challenges in Content-Centric Networkings (CCNs).

We close this dissertation with our overarching conclusions and a discussion of future research areas.

1.4 Previously Published Material

This dissertation draws on large parts of previously published work and writing that have been created jointly with several collaborators. We will indicate this at the beginning of each respective chapter.

2 Fundamentals and Definitions

Along the lines of the research presented in this dissertation, we introduce fundamental networking concepts for this work, beginning with an overview of underlying networking protocols relevant for streaming applications, i. e., the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). We then continue with an overview of the two emerging networking concepts relevant for future video streaming systems presented in this dissertation: Software-Defined Radio (SDR) and Named Data Networking (NDN).

With this introduction on the fundamental networking characteristics, we describe particular protocols for video distribution on the Internet—in particular, push-and pull-based approaches as well as relevant aspects of video encoding related to this work.

Last, we give an overview of foundations for performance analysis in Internet-based Video Streaming Systems (VSSs). This includes a discussion of content-centric video quality measurement, followed by network and user-oriented Quality of Service (QoS) and Quality of Experience (QoE) metrics in the context of HTTP Adaptive Streaming (HAS) systems.

2.1 Networking Concepts

2.1.1 Networking Protocols

The layered architecture of the Internet, based on the core concept of *separation of concerns*[6], is a central success factor for its resilience and applicability in heterogeneous environments and domains–and ultimately the indisputable success of the Internet in general. It allows the transfer of diverse application data, from simple text-based protocols (e. g., Telnet), to high-definition multimedia content, as discussed in this dissertation, without changing the underlying transport mechanisms. While routing is handled using `IP`, as an abstraction to the underlying `MAC` and `PHY` layers, transport protocols offer varying levels of functionality for an applications' data transmission requirements.

In this section, the most widely used transport protocols TCP and UDP are introduced, with an emphasis on attributes relevant for data transmission in video streaming applications.

[6] *"This is what I mean by "focusing one's attention upon some aspect": it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect's point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously."*
—Dijkstra [Dij82]

Transmission Control Protocol

The Transmission Control Protocol (TCP), as defined by Postel [Pos81], is:

> "designed to operate reliably over almost any transmission medium regardless of transmission rate, delay, corruption, duplication, or reordering of segments."

According to this specification and its stated principles, robust and refined versions of TCP are implemented in all relevant, network-capable systems (e. g., Linux[g]); applications can rely on abstract interfaces for data transmission traversing in adequate rates between hosts across potentially heterogeneous transmission architectures, Internet Service Provider networks, and continents.

While the discussion of all components, along with the vast research conducted in this area, goes far beyond the scope of this thesis, we will briefly introduce one of the key components of TCP that allows its successful operation in a wide range of systems: Congestion Control (CC). Considering the fundamental differences in the characteristics between networked links, i. e., highly robust and fast transmission in data centers compared to ubiquitous 2.4 GHz WiFi-APs deployed in households with often low transmission rates, high delays, and packet losses, an end-to-end link must be regulated to cater to this wide range of requirements. Fundamentally, CC regulates the packet flow based on some function derived from an input of an observed metric on the current connection. Todays most widely used TCP CC implementations include:

TCP Cubic that adjusts the congestion window size (cwnd) based on a cubic function for high bandwidth utilization.

TCP (New) Reno based on the additive increase, multiplicative decrease concept for cwnd control, adding the concept of fast recovery to its preceding implementation TCP Tahoe.

TCP Vegas which builds on measurements of the round trip time (RTT) to adjust the cwnd.

TCP BBR by Google [Car+16], identifies an optimal operating point (i. e., cwnd) based on the Bottleneck Bandwidth and Round-trip propagation time (BBR). The proposed CC has been shown to improve QoE when deployed for data transmission in a Dynamic Adaptive Streaming over HTTP (DASH) architecture by YouTube.

Apart from flow regulation, TCP ensures that received data (i. e., data that is passed on from the transport protocol stack to the application layer) is free of errors as both, header and payload, are verified with a checksum field specified in the header (see Figure 2). These guarantees, provided by TCP, however, come at a cost. In-order delivery requires that in case a packet is lost or damaged, it has the be retransmitted before data from packets following in sequence are passed to upper layers. This process can induce *head-of-line blocking*. For the interested reader, Callegari et al. [Cal+14] provide a detailed discussion of TCP concepts and CC algorithms.

[g] https://github.com/torvalds/linux/blob/master/net/ipv4/tcp.c

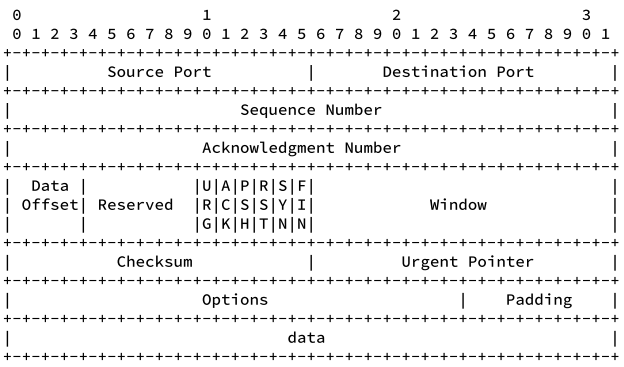


Figure 2: TCP Header

User Datagram Protocol

Given the description by Postel [Pos80], UDP is implemented

> "to make available a datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks [...] [and] provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed."

It is, therefore, concerning protocol overhead, a more lightweight alternative to TCP providing a high degree of flexibility for application layers but sacrificing guarantees about order and reliableness of data transmission, e. g., eliminating the problem if head-of-line blocking present in TCP.

Before the widespread use of HAS, video transmission protocols such as Real-Time Streaming Protocol (RTSP) relied mainly on UDP based transmission (however, this is not a mandatory requirement) as it offers fast, low overhead transmission of video data.

In recent years, UDP data has gained an increasing share of the total Internet traffic due to being used as a means to encapsulate higher layer protocols such as QUIC [Lan+17]; it is used by Google in conjunction with the Chrome browser and in the process of standardization by IETF.[h]

2.1.2 Software-Defined Radio

The term Software-Defined Radio (SDR) was first introduced in 1992 [Mit92]. It has since become a widely used concept that gained new significance in network research along with the success of Software Defined Networking and more capable hardware such as Field Programmable Gate Array (FPGA) and Application Specific

[h] https://datatracker.ietf.org/wg/quic/about/

Integrated Circuits (ASICs) [Ulv10]. In its most general definition, SDR is any radio interface that can be reconfigured by means of updating its software [Tut99].

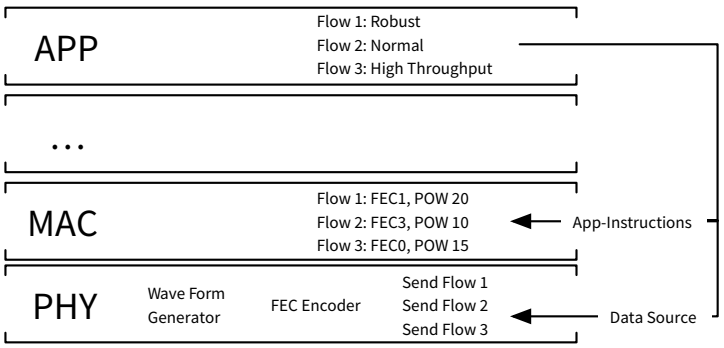


Figure 3: SDR Layers

One factor for their recent success is that SDR platforms now allow to re-program routines that were, in the past, only feasible when implemented directly in hardware in the form of ASICs. We classify these abstractions, for the discussion in this dissertation, into two areas, as shown in Figure 3: First, a large body of research is focussing on using SDR concepts to adjust PHY-layer waveform modulation, which allows for new application scenarios such as cognitive radio. Here, transmission adjusts dynamically to allow for context-aware adaptation of frequency spaces, modulation, and Forward Error Correction (FEC) depending on noise and cross-traffic. The second area, which has been the focus of research presented in this dissertation, is for SDR to provide North-bound interfaces to the higher layers, including applications, to receive current data transmission requirements. This approach makes it feasible to configure PHY-layer attributes so that they ideally support varying application needs.

Examples for SDR platforms are the WARP[i], relying on FPGAs, and the *nexmon*-Project [SWH17], that provides an API to implement routines on Broadcom Radio Interfaces reverse engineered on Nexus 5 (and other) devices.

2.1.3 Information Centric Networking

Information-centric Networks (ICNs) constitute a new paradigm in networking; content itself becomes the central entity while hosts are identified on-demand by the network. Various implementations of ICNs exist, such as DONA [Kop+07], SAIL,[j] and COMET.[k] In this work we will focus on NDN as this is the most researched and used implementation, that has also been used throughout the publications presented in this dissertation. For a comprehensive discussion of ICNs, we refer to Xylomenos et al. [Xyl+14].

[i] https://warpproject.org/trac
[j] http://www.sail-project.eu/deliverables/
[k] http://www.comet-project.org/

Named Data Networking

NDN is a complete implementation of the Content-Centric Networking (CCN) architecture proposed in [Jac+09] that is funded by the US Future Internet Architecture program.[1] Its primary design goals are to serve content location independent and highly available while guaranteeing security within a network. Communication relies on two types of packets: *Interest packets* request specific data objects while *Data packets* contain the content.

Any router containing the matching Data packet for an incoming Interest satisfies it by consuming the Interest and sending the Data packet to the user. Three data structures drive the forwarding engine in every router in the network: The Forwarding Information Base (FIB), the Content Store (CS) and the Pending Interest Table (PIT). Forwarding, handled by the FIB, relies on multicasting of requests for improved efficiency. The group of candidates for forwarding relies on identifying a matching hierarchical prefix, such as:

/tu-darmstadt.de/videos/bbb/mpd

These forwarded interests are stored in the PIT so that, in case of repeated requests for the same data, they can be fulfilled without reissuing that request to other nodes. Apart from eliminating duplicate and obsolete data requests, the PIT, represented on every node as the Interest requests' route, provides the back-propagating route to the consumer for a potentially following Data reply. Last, the CS caches previous Data replies, as specified be the given caching algorithms, such as First-In-First-Out or Least Recently Used. However, given that all passing Data messages are stored, the storage may fill up quickly. Thus, interaction with CS nodes based on a given data prefix may still be necessary as this provides storage of specific data on a larger scale. Yet, in case of packet loss and high request frequencies of Data, the in-built CS may provide improved network performance and efficiency.

[1] http://www.nets-fia.net/

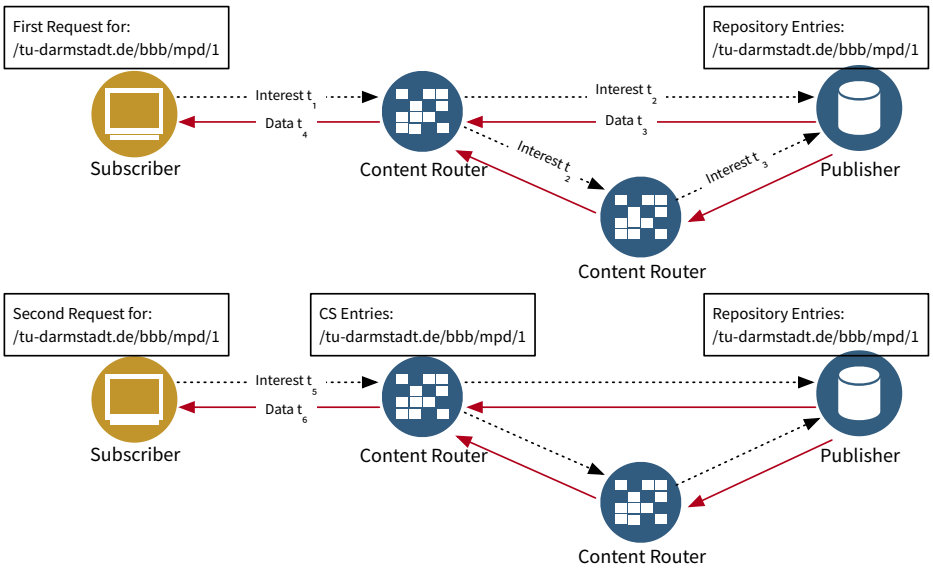


Figure 4: NDN request architecture

2.2 *Over-the-Top Video Delivery*

Following the previous introduction of underlying network mechanism (i. e., *Under-the-Top* of the application-layer), we begin with an introduction of the emergence of video streaming on the Internet, classified as Over-The-Top (OTT) services. In contrast to OTT, managed streaming services assume that a provider, such as Deutsche Telekom, has full control of the (end-to-end) network. While this enables better control and service guarantees, it lacks the flexibility of being usable in any network, regardless of the service provider. Another important difference is that, in contrast to interactive video communication where delays under 100 ms are required [SN95], OTT Video on Demand (VoD) has no hard realtime requirements.

We begin the analysis bottom up, starting with the encoding of video streams, transport streams, followed by push-based non-adaptive-streaming, and last pull-based adaptive streaming.

2.2.1 *Video Codecs*

An integral part of the success in Internet-based video delivery is the ongoing improvement of video codec efficiency that allows the compression of video content to a size feasible for transfer while maintaining a high degree of fidelity. This compression is defined by Richardson [Ric11] as:

> "[…] the act or process of compacting data into a smaller number of bits. [] Compression involves a complementary pair of systems, a compressor (encoder) and a decompressor (decoder). The encoder converts the source data into a compressed form occupying a reduced number of bits, prior to transmission or storage, and the decoder converts the compressed form back into a representation of the original

video data. The encoder/decoder pair is often described as a CODEC (enCOder/DECoder)"

For efficiency, all codecs used in OTT streaming relay on lossy compression as it greatly improves the potential for file size reduction compared to lossless compression (e. g., `huffyuv`) while visual quality for human observers is maintained [CPW11]. This relation is, however, depending on a multitude of factors including compression level (target bitrate), resolution, frame rate and the video content itself. An example of this relation, exemplified on the former two factors, is depicted in Figure 5.

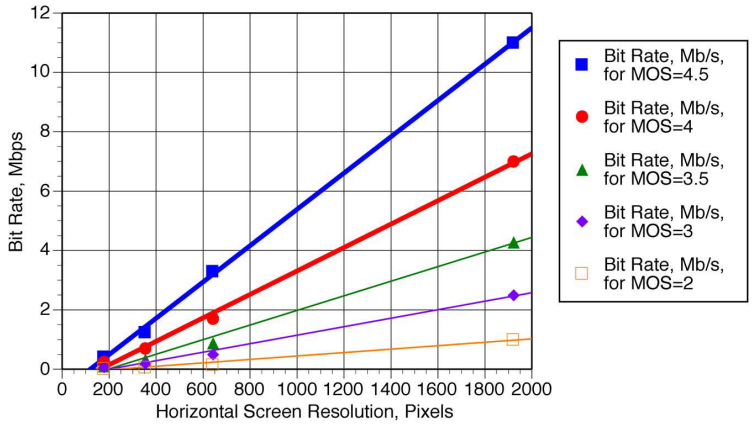


Figure 5: Relation between bitrate, resolution and subjective quality of video sequences encoded in H.264 by [CPW11]

From an abstract technical perspective, all major modern codecs employ similar techniques in encoding to achieve this: i) Exploiting discrepancies in the Human Visual Systems (HVSs)' capabilities to recognize differences with respect to color and structure; ii) translation to the frequency domain (e. g., by applying a Discrete Cosine Transform) and adaptive quantization; iii) inter- and intra-frame prediction to efficiently reuse visual similarity regions (i. e., instead of independently compressing each frame, regions that have a high similarity are encoded by referencing prior encoded regions and encoding only the delta); iv) last, general (lossless) compression techniques based on improved encoding efficiency (i. e., Variable-Length Codes) are employed to reduce the size of the data.

H.264/AVC

The by far most widely used codec today is H.264.[7] It significantly improves efficiency compared the previous standards leading to this widespread adoption since its standardization in May 30, 2003. The codec is widely supported in hardware. Especially for mobile video applications, this hardware support translates to less power usage for both, de- and encoding. Chip vendors, such as Qualcomm,[m] AMD,[n] and Intel,[o] provide APIs to expose H.264 codec functionalities as efficient

[7] *The full name of the codec is MPEG-4 Part 10, Advanced Video Coding (MPEG-4 AVC)*

[m] https://developer.qualcomm.com/forums/hexagon-dsp-sdk/video
[n] http://developer.amd.com/wordpress/media/2013/06/2904_2_final.pdf
[o] https://software.intel.com/en-us/media-sdk

hardware routines. This reduces power consumption compared to the regular software compiled using Reduced Instruction Set Computer or Advanced RISC Machine routines instruction sets. At the same time, H.264 maintains good levels of visual quality in lossy encoding. Last, the codec has been optimized for varying use cases, including real-time encoding to allow minimal time overhead for time-critical applications such as Mobile Video Broadcasting Services (MBSs).

2.2.2 *Scalable Video Coding*

One of the major drawbacks in today's adaptive video streaming architectures is the large degree of redundancy when encoding the same content in distinct layers to modify temporal, spatial, or quality attributes for HAS services. Generally, the original content is the same for each of these representations, and fundamentally these similarities could be used for more efficient encoding—much like within and between frames for independent representations. Scalable Video Coding (SVC) describes this concept and aims to reduce the inefficiency of Advanced Video Coding (AVC) by using layer similarities. To this end, a stream is divided to a base layer containing the lowest overall fidelity in all used dimensions, and one or more enhancement layers, each improving attributes that provide additional information for enhancing temporal, spatial, or quality dimensions upon the base layer. Even though SVC is, in theory, a superior concept to AVC, it has not been widely adopted as an extension to the H.264 codec. While the underlying reasons for this are not entirely evident, SVC does introduce additional complexity in the encoding and decoding process.[8] This translates to a set of software features required, and thus additional costs, that may hinder a wide usage.

[8] *"Worse is better" —Gabriel [Gab91]*

2.2.3 *Emerging Codecs*

Since the inception of H.264/AVC many new standards have been developed, e. g., H.265 by MPEG and VP9 by Google. These codecs generally leverage the same principles as described in Subsection 2.2.1 but improve prediction accuracy and compression efficiency, yielding a higher total degree of compression and better visual fidelity.

Driven by the desire to create a royalty-free codec with superior performance to H.265 (by MPEG) the *Alliance for Open Media*[9] has formed to standardize *AV1*. Its specification has been finalized on process.[p] While a discussion of possible new features to be included in AV1 is beyond the scope of this work, some notable improvements are made in the intra-frame prediction descriptors, the motion vector coding that can reference up to 7 frames, and a larger set of block sizes (up to 128 × 128 pixels).

[9] *Its members include Amazon, Apple, ARM, Cisco, Google, IBM, Intel Corporation, Microsoft, Mozilla, Netflix, and Nvidia*

2.2.4 *Push-based Real-Time Streaming*

With the emergence of consumers' use of video transported over the Internet, push-based streaming such as offered by RTSP [10] has been initially the main technical driver behind video delivery.

[10] *Adobes proprietary RTMP has been widely used in early 2000 for video delivery within flash containers*

[p] https://aomediacodec.github.io/av1-spec/

Real-time streaming mechanisms, as standardized by the RFC, are represented in multiple protocol specifications that can be used in conjunction (i. e., RTSP and RTP) and overlapping functionalities (i. e., RTSPs and Real-Time Control Protocols (RTCPs) monitoring functionality). First, RTSP [SRL98] is

> "[…] an application-level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video. Sources of data can include both live data feeds and stored clips. This protocol is intended to control multiple data delivery sessions, provide a means for choosing delivery channels such as UDP, multicast UDP and TCP, and provide a means for choosing delivery mechanisms based upon RTP."

Thus, as a second component, RTP provides [Sch+96]

> "end-to-end network transport functions suitable for applications transmitting real-time data, such as audio, video or simulation data, over multicast or unicast network services. RTP does not address resource reservation and does not guarantee quality-of-service for real-time services."

as an extension of RTP, RTCP standardizes additional feedback and status protocol mechanisms that allow server-side monitoring and control of the client's sessions to allow for delay-critical applications.

Yet, this server-side control and view on session characteristics induces a high computation overhead–it lacks scalability and context awareness on the client. I. e., a significant limitation is that a clients' context information has to be delivered and interpreted on a sending server, inducing a delay of at least one RTT. With larger numbers of clients, the management overhead on the server-side becomes critical.

2.2.5 Adaptive Streaming over HTTP

Driven by the limitations of push-based protocols, e. g., a high server-side management overhead and, firewall traversal issues, pull-based streaming approaches over HTTP have a become a successful alternative. As members of the class of HTTP Adaptive Streaming (HAS), many commercial implementations exist today, e. g. Microsoft Smooth Streaming (MSS), Adobe HTTP Dynamic Streaming (HDS), and Apple HTTP Live Streaming (HLS). A successful competitor to these commercial solutions is the open standard Dynamic Adaptive Streaming over HTTP. It is widely used in productive contexts on the Internet (see DASH.JS, Google Shaka Player, Bitmovin Player), as well as the de-facto standard in research. Hence, a detailed discussion of DASH is given in this thesis, exemplifying the characteristics of HAS following a brief description of notable differences in commercial players.

Dynamic Adaptive Streaming over HTTP

The DASH protocol has been adopted by most large streaming providers such as Netflix and YouTube (since Tuesday 27th January, 2015 as the default [Mon+17]),

as it provides a standardized protocol design with a high degree of flexibility for adaptive video transmission using the Hypertext Transfer Protocol (HTTP). Key characteristics for this flexibility are a modular design that relies on existing and common infrastructure elements used in the Internet such as regular Web Servers and Extensible Markup Language, as well as being codec agnostic. In summary, the key characteristics are: *i)* Support for video bitrate adaptation to enable use in diverse network scenarios; *ii)* a high design flexibility for adaptation choices so providers can cater to specific customer requirements and use cases; *iii)* client-driven adaptation for faster reaction to network changes with no required server-side state;[11] *iv)* can be fully integrated into existing HTTP-based distribution infrastructures including Content Delivery Networks (CDNs) and caches. Yet, some drawbacks exist with regard to this high flexibility, e. g., a high encoding overhead: Each layer has to be encoded separately. Further, as DASH is intended for use on the TCP/HTTP stack, multiple, mutually unaware control loops exist. Whereas video streaming using UDP-based protocols exhibits artifacts in case of packet losses, in case of DASH, there is a guarantee that packets are reliably re-transmitted in order. However, this instead may lead to stalling all following packets are delayed until the retransmit was successful. The effect is called *head-of-line blocking*. From the perspective of a DASH Adaptation Algorithm (AA) this is only observable as a lower bitrate.

From a technical perspective, DASH relies on a standardized Media Presentation Description (MPD) format, as shown in Listing 1. With regard to Figure 6 the main components of a DASH streaming system are:

[11] *This is, however, related to how such measurements are obtained. Segment request measurements become only available after each finished download. An exception is an approach used by YouTube that relies on byte range requests [Mon+17]. Newer web standards, such as the Network Information API[q] may allow improved feedback mechanisms in the future.*

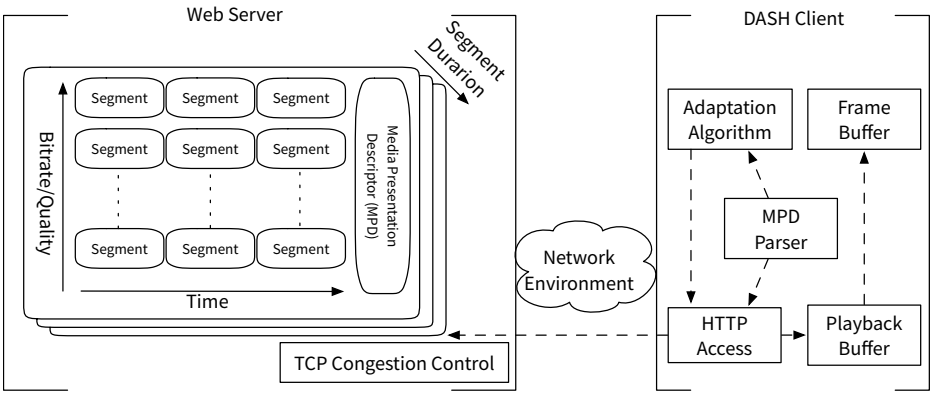


Figure 6: DASH architecture

Web server Repository of video segments exposed by HTTP protocol. Crucial performance aspects include the TCP CC.

Video content Representation of content encoded in discrete layers that may vary aspects such as Constant Rate Factor (quality adjusted by quantization), resolution, frame rate. Other parameters may exist, depending on the selected codec.

[q] https://developer.mozilla.org/en-US/docs/Web/API/NetworkInformation

Listing 1: DASH MPD example

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:dash:schema:mpd:2011"
  xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011 DASH-MPD.xsd"
  type="static"
  mediaPresentationDuration="PT654S"
  minBufferTime="PT2S"
  profiles="urn:mpeg:dash:profile:isoff-on-demand:2011">

  <BaseURL>http://example.com/ondemand/</BaseURL>
  <Period>
    <!-- English Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.40.5" lang="en"
         subsegmentAlignment="true" subsegmentStartsWithSAP="1">
      <Representation id="1" bandwidth="64000">
        <BaseURL>ElephantsDream_AAC48K_064.mp4.dash</BaseURL>
      </Representation>
    </AdaptationSet>
    <!-- Video -->
    <AdaptationSet mimeType="video/mp4" codecs="avc1.42401E" subsegmentAlignment
         ="true" subsegmentStartsWithSAP="1">
      <Representation id="2" bandwidth="100000" width="480" height="360">
        <BaseURL>ElephantsDream_H264BPL30_0100.264.dash</BaseURL>
      </Representation>
      <Representation id="3" bandwidth="175000" width="480" height="360">
        <BaseURL>ElephantsDream_H264BPL30_0175.264.dash</BaseURL>
      </Representation>
      <Representation id="4" bandwidth="250000" width="480" height="360">
        <BaseURL>ElephantsDream_H264BPL30_0250.264.dash</BaseURL>
      </Representation>
      <Representation id="5" bandwidth="500000" width="480" height="360">
        <BaseURL>ElephantsDream_H264BPL30_0500.264.dash</BaseURL>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Network context Delivery medium in OTT streaming that exhibits varying degrees of performance.

DASH client High-level classification of a software component providing stream-based playback of a video dataset, represented by an MPD descriptor. Video segments are requested from a web server, using the HTTP-protocol, across a given network context to retrieve a suitable video content as requested by a user. The selection of segments for download depends on current AA decisions, user input (i. e. seeking, pause events), and target buffer level. Retrieved video segments are ordered and written to a frame buffer, e. g., via the `MediaSource API`[r] in case of browser-based playback in Chrome.

Adaptation algorithm Software component responsible for selecting a given video representation to download based on measurement inputs derived by the `HTTP Access` component, `Playback buffer`, or other external input.

[r] https://developers.google.com/web/updates/2011/11/Stream-video-using-the-MediaSource-API

MPD Parser Interpretation of available video content based on the MPD requested from the web server.

HTTP Access Software component providing access to a given web server using the HTTP protocol. Additionally, it exposes statistics of download size and duration.

2.3 User-Generated Video

The last ten years have given rise to a vast adaption of smartphones, their connectivity and capabilities. This has enabled new use cases–in particular for video recording and distribution. Driven by this development, User-Generated Video services are provided by most large Internet companies, such as Facebook, Twitter, and Google. For the following chapters of this work, it is important to distinguish between the varying forms of those systems; along the areas depicted in Figure 7 we differentiate between five forms of User-Generated Video systems:

Mobile Video Broadcasting Services deliver a single stream of user-generated video in a direct fashion, i. e., live, to requesting viewers. The delivery may entail reencoding and distribution steps, depending on the delivery requirements (e. g., number of viewers, delivery network, real-time requirements).

On-demand User-Generated Videos describes the upload and storage of video sequences uploaded by users for VoD delivery.

Live collaborative User-Generated Videos describes an extension to Live-MBS that allows the composition of more than one source to a single User-Generated Video stream.

Personalized live collaborative UGV extends the above-mentioned approach by composing the User-Generated Video streams specifically to the demand of single viewers.

Automatic remixing systems provide collaborative video streams in an offline fashion; different sequences of User-Generated Video are uploaded and, after a defined minimum of video material is available, are analyzed to be composed into a single video sequence distributed to viewers as VoD content.

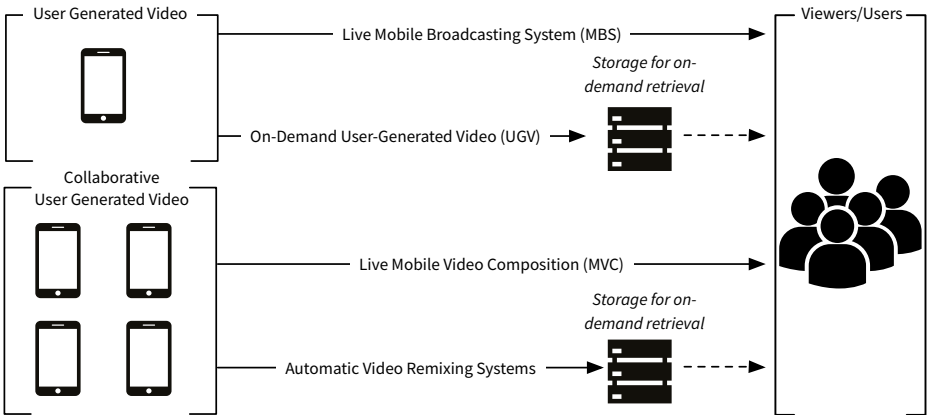


Figure 7: Overview of different User-Generated Video systems

2.4 Performance Evaluation

A prerequisite for improving QoE in a VSS is the ability to derive an objective understanding of the system. However, as QoE is fundamentally defined by users' subjective perception of a service (see Subsection 2.4.2), additional challenges are involved in the objective *performance evaluation* of VSSs.

In that regard, this section begins with the background for performance evaluation of video streaming services based on the fundamental networking concepts discussed in Section 2.1. Here, we give an overview of video quality metrics and then discuss aspects related to monitoring the streaming quality using QoS metrics, including delay, bandwidth measurement, jitter as well as video quality metrics. Building upon this discussion, we introduce the concept of QoE. Here, we introduce a general definition of QoE and its relation user studies (in particular using crowd-sourcing). Next, we introduce models, as introduced by the ITU, for the derival of QoE metrics using QoS measurements. It is important to note that the concepts discussed in this section building upon each other, as depicted in Figure 8. For a discussion of video performance evaluation concepts, focusing on content central analysis for User-Generated Video, we refer to a comprehensive overview by Wilk [Wil16, Chapter 2].

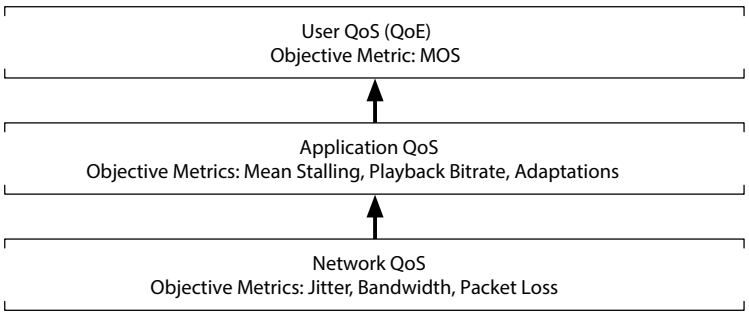


Figure 8: QoS/QoE Overview

2.4.1 Quality of Service

Different definitions of QoS exist, which are, in case of some sources, overlapping with QoE concepts. As we consider QoE as building upon QoS, i. e., a low bandwidth in a VSS may hinder adequate service delivery; there is, however, no direct causal relation between a single QoS metric and QoE. Thus, we refer to the definition of the IETF [Cra+98] for a definition of QoS as:

> "A set of service requirements to be met by the network while transporting a flow"

In this sense, based on the example given above, one of the requirements in a streaming application related to a sufficient bandwidth in order to transport a stream of a given bitrate. But, a discussion of service quality, requires to consider other aspects of the VSS, such as the concrete content delivered. Thus, the QoS concept does not directly qualify the influence of a given metric on the users' perception. Yet, different areas of QoS metrics exist, as discussed by Mok et al. [MCC11]. These metrics range in more abstract measurements, such as jitter, to application specific ones, such as the playback bitrate. Intuitively, the latter metric can more easily be related to a users' perception of a video streaming service. However, a fundamental drawback is that these metrics do not take into account the content itself. Thus, we also briefly introduce objective video quality metrics in the following.

Network QoS

Packet loss: The number of received packets, nP_r, over all sent nP_s, is defined as packet loss rate in a data transmission, given $L = 100 * \frac{nP_r}{nP_s}$ [XWP03]. This loss of data can is either directly detectable on the transport layer, e. g., when TCP is used and segment numbers are available. When loss is not detected on the application layer, loss recovery is still possible on the application layer (e. g., by means of FEC). Yet, in video streaming applications, especially with real time requirements, it is often not desirable as it can increase delay and jitter, i. e., by inducing head of line blocking [MPF16].

Jitter: describes a measure of the variance in delays between successive packets [XWP03]. In video transmission, high negative effects of jitter are usually mitigated by larger playback buffers.

Bandwidth: is a measure of the average utilization of a link for a given time period [Pro+03], e. g., the fraction of the used capacity of a link. In relation to video streaming, this measure is an upper limit for the mean achievable playback bitrate. If the mean bandwidth remains below the playback bitrate, the playback buffer depletes–thus stalling will occur. For a bandwidth larger than the playback bitrate, the buffer fill state will increase. The bandwidth estimate as calculated on downloaded video segments is used as a basis for the selection of future video segment bitrate in Throughput-based Adaptation (TBA).

Application Quality of Service

Stalling: One of the most detrimental effects on the users' experience is *stalling* [Seu+15]. A video playback session is considered to be in a stalling state if, after the playback has started, the buffer becomes fully depleted–before the end of the video–and as a result playback must be interrupted. The duration of the video stall usually lasts until the buffer has been refilled to a minimum desired buffer fill level (this value often depends on the given implementation used). There are two main stalling metrics. First, the stalling frequency ν_s and second, the stalling duration D_s where the total stalling duration of a video is $\sum D_s$ for all stalling events $s \in S$.

Initial Delay: Unlike stalling, the initial delay describes the time until the initial playback starts after the player is launched. Here, generally, a smaller value is desirable as long waiting times may lead to churn of users. However, this low initial waiting time requires to start with a lower video representation, as these segments are usually smaller in file-size and lead to a higher startup delay [Bal+12]. Thus, for achieving a high playback bitrate, further adaptation steps become necessary.

Playback Bitrate: A key metric in adaptive streaming is the playback bitrate. It provides an approximate measure of the playback quality at any given time t, R_t or the mean bitrate ϕ_{R_T} of a session $\frac{1}{T} * \sum_{t=0}^{T} R_t$ [Seu+15]. However, some limitations exist in this metric. The relation to quality strongly depends on the used codec and its parameters, the content as well as the respective sequence of the encoded content. As most modern codecs, such as H.264, perform best using adaptive bitrate encoding, thus using more bytes for complex scenes and fewer bytes for simpler scenes as a strategy to increase to overall perceived quality of a video sequence, the actual size of segments can also vary widely (see Figure 9) [WRZ16]. As a result, the mean bitrate usually specifies an approximation of the actual transferred bytes and can vary depending on which segments layers have been selected for each given segment and its corresponding size.

Adaptations: As stalling and the mean playback bitrate can be expressed as a simple optimization problem, i. e., maximize the mean playback bitrate under the condition that stalling = 0. However, it is also important to consider the effect of adaptations in adaptive streaming. It describes a measure of the switches between layers. Rodriguez et al. [Rod+14] examined the effect of Video Representation Switches (VRS) showing that the frequency of representation switches impacts the viewers' subjective perception significantly. A related metric is the magnitude of these switches [Zin+03]. The larger the jump, the more perceivable a switch is to the user. It can be shown that many switches with a low magnitude are better than a single switch with a high magnitude, such as described by Wilk, Stohr, et al. [WSE16].

For a further discussion of the QoS concepts, we refer to [GJS03].

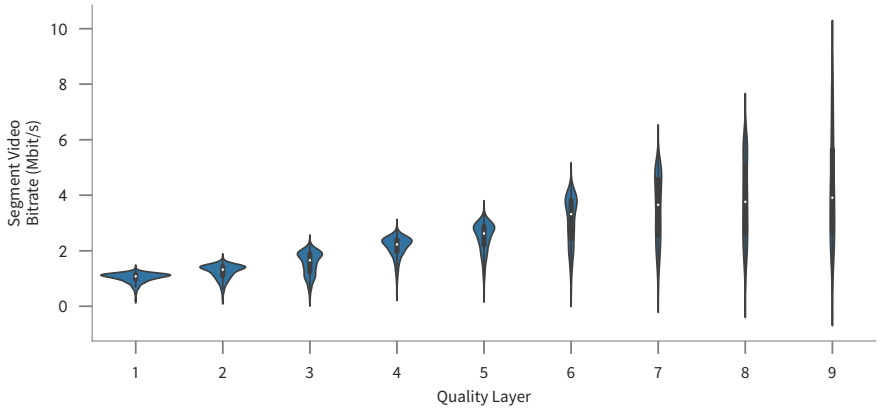


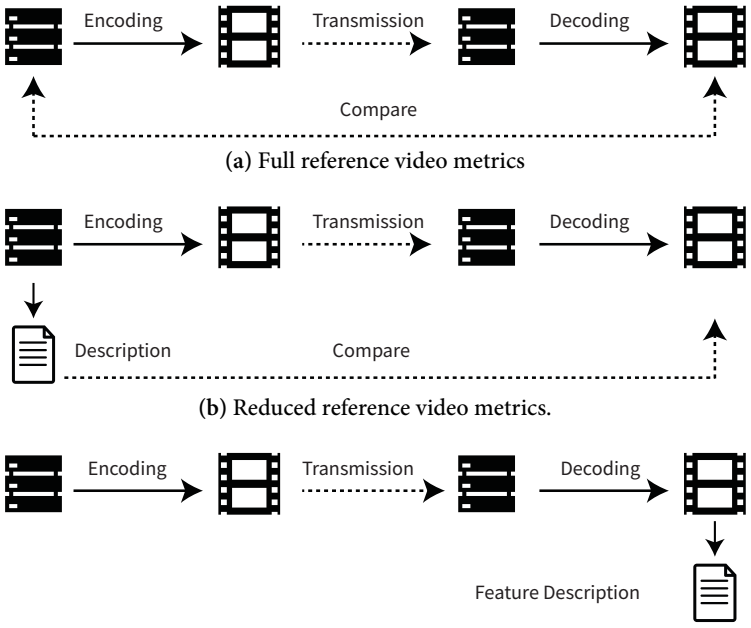
Figure 9: Variation of segment sizes for the Big Buck Bunny video sequence encoded in 2-second segments using H.264. Respective shapes indicate representations. Inner black boxes represent median and IQR, with extending lines indicating the full distribution span. Further, the blue areas indicate this distribution as a kernel density estimate.

Video Quality Metrics

Objective video quality assessment can be categorized into three areas, *1) no-reference-*, *2) reduced-reference-*, *3)* and *full reference-* analysis, as depicted in Figure 10. Each approach comes with specific advantages and disadvantages that promote their use in specific scenarios. We will briefly discuss these three areas of video quality assessment, along with the most prominent concepts presented in research.

Full Reference Video Metrics: The most widely used concept to evaluate the fidelity of videos sequences are full reference metrics. Generally, these metrics work by directly comparing the original video (or signal) to the degraded signal. The most fundamental concept here is Mean Squared Error (MSR). It directly evaluates the mean of the absolute distance between each pixel in each channel in their corresponding location. Based on this concept Peak Signal to Noise Ratio was introduced by Stathaki [Sta11] which is derived from the MSR to express to relative noise in a video signal in dB. The Structural Similarity Index (SSIM) extends the basic approach by including characteristics of the HVS, thus providing a better correlation between the metric and the evaluation by human observers [Wan+04]. One of the most recent approaches that extends this idea, developed by Netflix [Li+16], is called Video Quality Model with Variable Frame Delay (VMAF). It applies machine learning concepts to derive a measure with improved correlation between various artifacts in a video sequence and the resulting influence on the subjective experience of a human observer.

Reduced Reference Video Metrics: Whereas full reference metrics rely on the original video source, which in many cases has a considerable file size, reduced reference metrics use some form of derived information based on the origi-



(a) Full reference video metrics

(b) Reduced reference video metrics.

(c) No reference video metrics.

Figure 10: Overview of objective video quality assessment

nal content as an input for their evaluation function (see Figure 10(b)). The main advantage here is that this derived information can be much smaller than the original content (e. g., a basic numeric descriptor or scalar per frame) and thus makes this option feasible for usage scenarios were the transmission and access of the original video content is not an option.

No Reference Video Metrics: In contrast to the approaches introduced above, no-reference metrics work independently of the original video content [WM08]. Traditionally the evaluation of content fidelity has been very challenging in as there is no reference to the original content that can be used as a comparison to evaluate the loss of fidelity. However, recent advances in machine learning provide the means to derive approximations of a video sequences fidelity by incorporating models that relate to the human perception (e. g., [Kan+14]).

2.4.2 Quality of Experience

The general idea of measuring human satisfaction with a given service—the QoE—and thus providing a measure beyond pure system performance and including humans perceptive, has been discussed since 1990 and emerged as a fundamentally more relevant concept compared to QoS. As a general definition, Brunnström et al. [Bru+13] describe QoE as:

> "[...] the degree of delight or annoyance of the user of an application or service. It results from the fulfilment of his or her expectations with respect to the utility and / or enjoyment of the application or service in the light of the user's personality and current state."

This is by no means the only existing interpretation of this concept; A multitude of slightly varying and evolving definitions can be found (e. g., by the International Telecommunication Union (ITU) [ITU08]).[12]

[12] *This QoE concept can nowadays be found in a wide range of use cases and disciplines such as Web design [Hoß+11; Iba+09] and mobile Networking [SLC07; Zhe+16].*

Based on the previous discussion of how QoE relates to users and preceding system-oriented measures, we argue that for *User-Centric Video Streaming*, QoE is a fundamental concept and therefore of high relevance within this dissertation. Here, in particular for OTT HAS, QoE oriented assessment became a central target for studies since the streaming process itself has an impact on the perceived quality.

Mean Opinion Score

The gold standard for the assessment of QoE are user studies. The most widely used concept for the evaluation is the Single Stimulus Continuous Quality Scale (SSCQS) scale, as recommended by the ITU. This continuous scale, as depicted by an example in Figure 11, is used to gather feedback on the subjective impression for each independent use. The MOS is derived by calculating the arithmetic mean over all N assessments for each subjects' (i) scores $v_{i,j}$ of a given video j :

$$MOS_j = \frac{1}{N} \sum_{i=1}^{N} r_{i,j} \tag{1}$$

A detailed discussion of other steps involved, e. g., data inspection and normalization, we refer to ITU-T [ITU12].

Newer concepts in conducting user studies have been more adopted recently that reduce the overhead required for these experiments. In crowd-sourced user studies, test subjects are recruited using online platforms e. g., Amazon Mechanical Turk[s],[13] Microworkers[t], and Crowdee[u]. This method has the advantage that more participants can be reached to allow higher statistical confidence, and can be initiated ad-hoc, usually with far less financial resources. Drawbacks do however exist as, in conducting experiments, external factors such as the system and device setup are much harder to control, and variation in results is often higher [Hoß+14a].

[13] *This name refers to the first artificial chess player, albeit this was in fact operated by a human chess player hiding in the apparatus.*

Just Noticeable Difference

As variance, given the inherent *individual* differences in humans and their changing mental states, is commonly high in MOS-based QoE assessment, an alternative concept is an evaluation on the grounds of a direct comparison between sequences—the Just Noticeable Difference (JND). This concept is also used in conjunction with VSS studies to obtain a statement regarding the preference relative to two stimuli.

Quality of Experience in HTTP Adaptive Streaming

A critical challenge for QoE in HAS is to derive models that incorporate QoS factors as well as their interdependence with regard to a measure for QoE that

[s] https://www.mturk.com/
[t] https://ttv.microworkers.com/index/template
[u] https://www.crowdee.de/en/

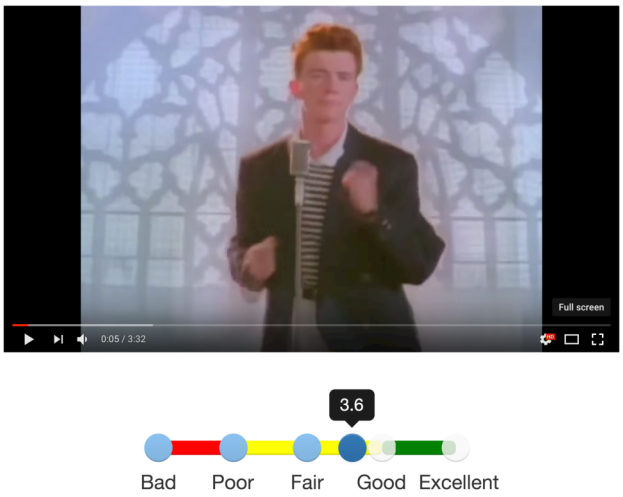


Figure 11: Example for an SSCQS scale implemented as an interactive web component to be used within user studies.[w]

provides a right balance between bias and variance. Hence, measured QoS depends on the relation of the user to the current video sequence; however, their influence on QoE can be generalized to some degree. For example, low-bitrate playback is less relevant to a user for a low structure scene, such as video credits, as compared to high structure scenes.

Given that QoE is the ultimate measure of performance in HAS, robust estimation of MOS from QoS metrics is an important research topic. Here, it is the most common method to derive such models with machine learning concepts such as regression-based modeling. One of the key influence factors for deriving QoE estimates is stalling. A stalling-based QoE metric was published in [Hoß+13], featuring duration and frequency as input for a regression model. Likewise, the perceptual models by the ITU [P1213] use stalling duration, frequency, and the initial delay to derive a MOS scaled *buffer-related perceptual* QoE metric. In order to quantify stalling effects during playback, Hoßfeld, Schatz, et al. [Hoß+13] introduced a model to estimate MOS from the stalling frequency and average duration based on user studies for YouTube.

QoE models based on playback quality exist, but they actively depend on many factors such as the relation of the compression method (its configuration), resolution, the numbers of quality layers, as well as the content itself. Hoßfeld, Seufert, et al. [Hoß+14b] quantified the impact of VRS. The authors found that the time spent on the highest representation layer has a more significant impact on the MOS than the frequency of switches and introduce a QoE model based on their findings.

[w] https://gist.github.com/577c522a308faa969767e8d492a57093

3 Related Work

We structure the related work along the three main research areas of this work, namely mobile video upload, as well as interdependencies and future networking concepts in Dynamic Adaptive Streaming over HTTP services.

3.1 Mobile Video Upload, Distribution, and Composition

First, this section describes related work with a focus on investigating the mobile video distribution, upload, and composition of User-Generated Video (UGV). Beginning with an overview of work analyzing Mobile Video Broadcasting Services as well as Live-Video Streaming Systems, we introduce concepts that incorporate aspects of Quality of Service (QoS) and Quality of Experience (QoE) evaluation within Mobile Video Upload and Mobile Video Composition (MVC) systems for live video composition that correspond to the ideas presented in this dissertation.

3.1.1 Mobile Video Broadcasting Services

A detailed analysis of TCP and the behavior on the streaming sessions has been conducted by Alcock et al. [AN11]. They analyzed streaming sessions for YouTube and Netflix mobile clients in WiFi connections. Regarding data transfer, YouTube Android clients do usually download an initial amount of data ranging from 4 –8 MBytes starting with an initial burst phase of data transfer. As the unrestricted downloading of video segments in fast mobile networks could result in a waste of data volume in case the session is stopped preemptively, long `ON-OFF` transmission patterns have been observed, regularly stopping data transfer and thus shaping the download speed when the buffer reaches a given threshold. Here, in contrast to Android, iOS devices have only short `ON-OFF` patterns, using more than one TCP connection Alcock et al. [AN11].

Ramos-Muñoz et al. [Ram+14] investigated usage characteristics for YouTube on Android and iOS devices in 3G cellular networks. Here, the apps use HTTP range requests for DASH and Apple HTTP Live Streaming (HLS) respectively and establish multiple TCP connections in a session. Even though they mainly use one connection for over 90% of their download, having multiple TCP connections increases fault tolerance. Access to videos is controlled by YouTube's video web servers in a way that they allow an initial burst phase to download video segments at full speed, following the throttling of the transmission speed once the buffer has been filled. Regarding the encoding, at the time this analysis was conducted in 2014, H.264/AVC was mainly used in these services. Looking at more recent studies,

such as conducted by [Zha+17] in 2017 other–more efficient–codecs, including VP9, are also used now.

In Finamore et al. [Fin+11] the network traffic of users accessing YouTube over WiFi is analyzed. The Transmission Control Protocol (TCP) connections are used for an initial burst phase in which long ON-OFF transmission patterns have been observed, which are regularly stopping data transfer and thus are also shaping the download speed [AN11]. Regarding user characteristics, they have found that a high percentage of clients abort the streaming session before the end of the video.

A recent field study of YouTube traffic [Seu+17] investigates mobile device app use of Android users. Overall the study reveals the distribution of connections types, as depicted in Figure 12, that indicates that most users still use predominantly 3G connections. With regard to QoE aspects, it is shown that about 35 percent of all sessions experience some stalling during their playback. Furthermore, a generally low amount of adaptation steps indicates a more conservative Adaptation Algorithm (AA).

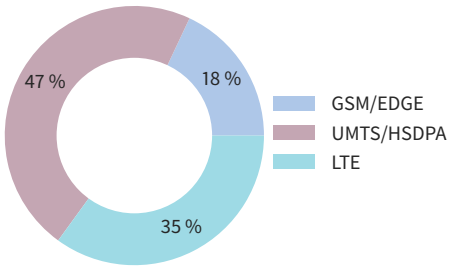


Figure 12: YouTube traffic by device type based on data obtained from the YoMoAPP study [Seu+17]

3.1.2 Live Video Streaming Services

C. Zhang et al. [ZL15] investigated how video streams are accessed and produced in the case of *Twitch.tv*, a live streaming platform for video game broadcasts. Twitch.tv uses Real-Time Messaging Protocol (RTMP) to stream video from the broadcasters to the servers and then transcodes the video for an HLS-based distribution. The authors report that Twitch.tv has, in peak situations, up to 12 000 parallel video streams. An end-to-end duration between recording and watching a video is on the average 21 seconds. This can be seen as a moderate to a low value for HTTP-based delivery of video over a Content Delivery Network (CDN). The popularity of content shared follows an extremely skewed Zipf distribution in which around 0.5% of the broadcast streams account for 70% of the views.

Pires et al. [PS15] explore Twitch.tv and state that usage peaks generate data traffic of 1 Tbps. Further, they compare Twitch.tv with YouTube Live, another live-streaming service, and show a significantly lower popularity in comparison to Twitch.tv; whereas Twitch.tv seems to deliver at any time more than 6000

channels/live streams, YouTube Live does accounts for around 300 to 700 channels (this study was conducted in 2015).

Regarding the design of Live Video Broadcasting Services, the work of El Essaili et al. [El +15] investigates the uploading of video under the assumption that the resource allocation in the used LTE network can be adjusted. They propose a centralized quality-oriented decision for the uplink transmission of the client-side video.

A queue-aware scheduling scheme for improved QoS in mobile uploads was proposed by Rizk et al. [RF16]. This approach can achieve lower overall queue lengths and delays for, e. g., live video uploads, as demonstrated in simulative emulations.

Seo et al. [SCZ12] discuss how DASH can be used for the upload of media by leveraging the Hypertext Transfer Protocol (HTTP) `POST` requests, in contrast to regular DASH where clients fetch segments using `GET` requests, to continuously upload video segments. The proposed system is able to transcode and transmit video with a restitution of up the 480p, and initials delays of approximately one segment duration.

Johansen et al. [Joh+09] propose a system designed to generate video segments and upload them immediately in order to generate a low-delay video streaming experience. This concept includes the adaptation of video bitrates during the streaming session to handle variations in the network conditions.

To leverage better scalability and efficiency on the encoding step, a system by Siekkinen, Masala, and Kamarainen [SMK16] achieves a better uplink utilization using Scalable Video Coding (SVC) and thus adapting to changing network conditions. Similarly, a system based on SVC content is proposed by Richerzhagen, Wulfheide, et al. [Ric+16]. Here, the authors propose a concept for source selection in collaborative video upload using data on the clients' context, i. e., the measured network bandwidth. For the upload of video on selected streaming sources, they suggest network resource sharing between nearby clients to address situations where the upload capacity is insufficient for the transmission of the desired video source. Theoretical limits of such collaborative uploads have been shown in [Khu+18]. The system's performance of [Ric+16], has been evaluated in a simulation, showing that with SVC encoded stream a continuous playback for low-bandwidth situations can be achieved, also in cases where direct upload strategies are not feasible.

Last, another recent approach for mobile video upload based on SVC has been proposed by [SMN17]. The authors define and evaluate, using a simulation approach, a DASH-based video scheduling heuristic for SVC encoded segments.

3.1.3 Video Composition

Along with the categorized view on User-Generated Video composition, shown in Table 1, the following section gives an overview of related work in this area.

One of the first video composition systems has been proposed by Engstrom et al. [EEJ08], considering video quality to compose a video mix. However, it is a semi-automatic system and therefore out of scope in this dissertation.

MoviMash [Sai+12] introduces an automatic composition, combining a video quality metric with an analysis of video recording degradations. The latter describes phenomena common in User-Generated Video, caused by the lack of skills of the recording user or technical limitations of the recording device. Their video composition algorithm combines video streams from different sources, analyzes them and neglects the views in which recording degradations occur.

A formalized automatic composition system is introduced by Shrestha et al. [Shr+10], offering an algorithm based on an objective quality function. It maximizes the quality of music video clips, integrating the completeness of the composed video, suitable cutting points, length of the individual shots, and diversity of the views.

In the work by Cricri et al. [Cri+12] a first concept for replacing computationally expensive video analysis is shown. They propose to leverage different sensors of the recording devices to compose the video based on this information.

Table 1: Overview of Video Composition Systems

System	Composition	Switch Delay	Analysis Methodology	Evaluation System
MoVi [BR10]	offline	–	**Content:** social device grouping; light intensity; view similarity; acoustic; **Context:**PoI	expert control
Pulse [Bao+13]	–	–	**Content:**: n/a **Context:** face tracking; lip tracking; blink-detection; sound; accelerometer; gyroscope; device interaction	expert control
Cricri et al. [Cri+12]	offline	–	**Content:**: n/a **Context:** compass; GPS; accelerometer	precision/recall/F-measure: manually annotated vs classification by system
MoViMash [Sai+12]	online	–	**Content:** blurriness; blockiness; contrast; illumination; views diversity; occlusion; **Context:** tilt angle; shakiness; gyroscope; accelerometer	user study (expert composition control)
LiViU [WE14a; WE14c]	online	20 seconds	**Content:** shakiness, misalignment, harmful occlusions **Context:** shakiness (accelerometer, compass)	user study (corwdsourced, lab) with dataset [Sai+13]
ContextNotContent [Sto+16c; Sto+17b]	online	$\backsim$ 5	**Content:** n/a **Context:** accelerometer, network probing, location, user actions	user study (random composition control)

A more recent work following this sensor-based approach for video composition is the *Automatic Video Remixing System* by Mate et al. [MC17]. It resembles the general concept of incorporating sensor information for User-Generated Video composition presented in our previous work [Sto+16c; Sto+17b]. While the presented work also includes simple content analysis such as brightness, content-aligned sensor information such as acceleration and GPS-based positions are recorded for single recording users. Further, this information provides clues on the PoI based

on collaborative data analysis conducted centrally. Here, the selection of a video stream is re-assessed continuously considering factors such as video quality or diversity of the final composition.

Last, with the *Live Video Upload System (LiViU)* by Wilk and Effelsberg [WE14a; WE14c] provides a system for live recording and upload of video streams from mobile devices. Its functionalities include the online *composition* and broadcasting using video content analysis for streamed live video content.

3.1.4 Discussion

With Cricri et al. [Cri+12] a first step towards replacing traditional video content analysis with mechanisms that leverage sensors of the recording devices to compose the video was introduced. The authors describe the main advantage to be a significantly reduced processing time. However, Cricri et al. [Cri+12] solely inspect the camera movement for composition decisions. Thus, network-related aspects and higher level context features are not considered.

The *Automatic Video Remixing System* by Mate et al. [MC17] has been evaluated with a focus on different event contexts. Yet, the actual the live composition of User-Generated Video is note considered as part of this work.

Last, while *LiViU* by Wilk, Zimmermann, et al. [WZE16] supports video-composition based on content analysis we are only building upon this general functionality in LiViU adding sensor-based composition concepts introduced in this work.

3.2 Network Interdependencies in OTT Video Streaming

In the following, we provide an overview of related work for the analysis of Over-The-Top (OTT) Video Streaming Systems. We begin with a discussion of work investigating DASH AA, followed by work that conducts cross-layer and large-scale parameter space studies.

3.2.1 Overview of DASH Adaptation Algorithms

An overview of the discussed AAs is given in Table 2. We categorize related work for DASH AAs in three main areas, namely: Throughput-based Adaptation (TBA), Buffer-based Adaptation (BBA) and other concepts that rely on external input. In each category, we present the most noteworthy approaches that have been used in the comparative studies conducted in this dissertation. We do not aim to provide an exhaustive overview of work on AAs as this would go beyond the scope of relevant related work. A recent survey that provides a comprehensive overview in this field was published by Kua et al. [KAB17].

Throughput-based Adaptation (TBA)

Throughput-based (or rate-based) Adaptation Algorithm rely on the estimation of the currently available network bandwidth for decisions on future video layers rates L. They compute the throughput of the present connection using a function

Table 2: A selection of significant publications on AA in DASH. Highlighted rows are investigated in the context of this thesis (TBA is Throughput-based Adaptation and BBA is Buffer-based Adaptation).

Algorithm	Year	Type	Buffersize	QoE Metric	Comparison	Evaluation Scenarios
BBA-X [Hua+14]	2014	BBA	240s	no	Smooth-Streaming	Netflix A/B Testing
FESTIVE [JSZ14]	2014	TBA	30s	no	Smooth-Streaming	Multiclient
PANDA [Li+14]	2014	TBA	min. 26s	yes	FESTIVE, Smooth-Streaming	Multiclient
BOLA [SUS16]	2016	BBA	25s	yes	ELASTIC, PANDA	Mobile traces
VAS [WSE16]	2016	External+ (Hybrid, BBA, TBA)		VQM ([PW04])	[Mil+12]	Mobile traces ([Eit+13])
SQUAD [WRZ16]	2016	Hybrid	30s	yes	VLC, SARA, BBA	longrun TCP, over Internet (US-GER)
PENSIVE [MNA17]	2017	Hybrid, External	60s	[SUS16; Yin+15]	BOLA, FESTIVE, BBA-1	LTE, Traces,
YouTube's AA [Mon+17]	2017	BBA	70s ([Año+18])	n/a	n/a	n/a

$f(R(s_{t,l}), ..., R(s_{t-n,l}))$ on the measured Rates R executed after the download of a segment S_l considering a sliding window of the n previous measurement periods. According to the estimated bandwidth, the quality level is adjusted. In a simplified view, the best choice for the quality level is the highest video bit rate that fits completely within the estimated bandwidth. Ideally, the bandwidth estimation function detects changes in bandwidth fast enough for the algorithm to react and adjust the quality level (i. e., before stalling events occur when the bandwidth drops heavily) while not introducing *adaptation hysteria* (i. e., frequent changes in the playback bitrate) caused by short-term bandwidth fluctuations. One notable example of such an algorithm in research is *PANDA* [Li+14]. It uses a *Probe and Adapt* scheme in reference to TCPs Additive Increase Multiplicative Decrease principle, to predict the bandwidth and to avoid an unfair share of bandwidth at network bottlenecks. Approaches used in practice, e. g., as implemented in Google's Shaka Player, rely on an exponentially weighted moving average (EWMA)[x] over the estimated throughput measurements and adapt when defined up- and down thresholds are crossed.[y] Until recently TBA was also used in the DASH.IF's reference player *DASH.JS*.[z,14]

[14] *As of version* `2.6.0`[aa] *DASH.JS switches the Adaptation Algorithm mechanisms dynamically between BOLA and TBA*

[x] https://github.com/google/shaka-player/blob/v2.3.5/lib/abr/ewma.js
[y] https://github.com/google/shaka-player/blob/v2.3.5/lib/abr/simple_abr_manager.js
[z] https://github.com/Dash-Industry-Forum/dash.js/blob/v2.1.1/src/streaming/rules/abr/ThroughputRule.js
[aa] https://github.com/Dash-Industry-Forum/dash.js/releases/tag/v2.6.0

Buffer-based Adaptation (BBA)

Buffer-based adaption algorithms use measurements of the DASH-players' buffer as input to derive the bitrate of a segment to download in the future—hence they are *buffer-based*. Generally, a consistently growing buffer level is an indication that the current playback bitrate is lower than what could be supported to be transmitted through the network. If the buffer level falls, the playback bitrate is too high considering the network conditions and must be reduced to avoid stalling. The video bitrate is appropriate when the buffer value is balanced, i. e., it remains constant within a given margin of fluctuation. This approach has first been described by [Hua+14] as part of Netflix's commercially used adaptation strategy.

In this work, it was shown that, at the time of writing, this approach outperformed TBA approaches after the startup-phase of the playback has passed.

A more recent BBA called BOLA, published by [SUS16] uses the buffer state as an input for a *Lyapunov* optimization to calculate the best joint utility that achieves a minimum of rebuffering along with a maximized video quality.

Other Adaptation Algorithm Concepts

A multitude of Adaptation Algorithms exist, that do not directly fall into either of the two categories mentioned above. One approach is the combination of both approaches to a hybrid concept that uses a bandwidth estimation function as well as the buffer fill level. Thus, it capitalizes on the advantages from both types of adaptation–along with disadvantages of both. One example of a widely cited hybrid AA approach is Spectrum-based Quality Adaptation for DASH (SQUAD), proposed by C. Wang et al. [WRZ16]. The architecture of SQUAD is based on a decision tree: for the startup phase or critically low buffer levels, throughput measurements are used. Otherwise, bitrate adaptation decisions are based on obtained measurements of a classification on the *spectrum* of TCP measurements. This step is conducted with the underlying assumption that, depending on the video segment size, TCP throughput varies.[15] It is important to note that SQUAD, unlike other adaptation concepts, assumes knowledge of all individual segments sizes. This is generally hard to achieve in live streaming scenarios[16].

Mondal et al. [Mon+17], by means of reverse engineering, investigate YouTube's current AA. Their findings indicate that YouTube uses primarily a BBA adaptation scheme, but, in contrast to other approaches, adapts both the video bitrate and the segment length of video chunks. The latter is used as a strategy to avoid stallings in case sudden bandwidth (i. e., throughput) drop situations occur. Also, for adaptation steps increasing the bitrate, YouTube's algorithm opportunistically requests both, the current bitrate and the target bitrate segments; for lowering the bandwidth only the target bitrate is requested, thus fully utilizing the already buffered video segments for playback.

Other concepts that fall in this third category rely on an integrated analysis of server and client-side metrics that derive decisions using machine learning models. One such approach has been recently introduced by Mao et al. [MNA17]. It uses reinforcement learning to train a neural network model for bitrate selection on the client. While the training and inference take place on the server, the systems'

[15] *This also relates to similar reasoning described in the downward spiraling effect discussed by Huang, Handigol, et al. [Hua+12] as presented in Subsection 3.2.2.*

[16] *Unless overall encoding efficiency is sacrificed by using fixed bitrate encoding. This is generally discouraged.*

input are client-side measurements that include "the current buffer occupancy, rebuffering time, chunk download time, size of the next chunk (at all bitrates), and the number of remaining chunks in the video." [MNA17]. Using a QoE centric evaluation function, the authors show a clear performance improvement over other approaches like BOLA as well as early BBA [Hua+14] and TBA [JSZ14] concepts.

Last, another server-supported concept—thus relying on external input for client-side decisions—has been proposed by Wilk [Wil16]. The general premise of this work is novel as in that, instead of following the indirect goal to improve QoS factors, i. e., maximize bit-rate while minimizing switches and stalling, [Wil16] suggests that QoE *with regard to the specific video sequence* is the direct optimization goal. To this end, an external service provides efficient full-reference video analysis for deriving a content-aware optimal adaptation plan that is used to improve the overall QoE in adaptation decisions given current bandwidth constraints observed by the player.

3.2.2 Network Interdependency Effects in DASH

In contrast to studies comparing the performance of DASH in a fixed set of conditions, e. g., a single bandwidth trace with varying DASH AA, this section presents work on the systematic performance assessment of DASH under the variation of *multiple* system conditions, including those underlying the DASH application itself. We refer to this as analyzing network *interdependencies* in DASH.

A first study on the interaction of DASH streaming and delivery networks has been published by Huang, Handigol, et al. [Hua+12]. Here, the authors discuss the interaction of DASH with TCP, which, in the case of network bandwidth drops, e. g., due to competing clients, leads to an effect called *downward spiraling*. It describes an ongoing decrease in playback bitrate after a single bandwidth drop due to the slow start behaviour of TCP combined with requesting smaller segments sizes by the player. For two novel TCP mechanisms, *newCWV*, and *newCWV*-pacing, Nazir et al. [Naz+14] have examined their performance in cross traffic and non-cross traffic scenarios, showing that *newCWV*-pacing improves performance compared to TCP NewReno in DASH streaming scenarios.

As part of a work proposing a prediction-based AA for mobile DASH streaming sessions, Evensen et al. [Eve+14] analyze the influence of TCP Congestion Controls (CCs) on the number of packet drops, as well as the influence of the segment size on the congestion window.

Next, Maki et al. [MVA16] relate DASH playback sessions' QoS factors to a set of configuration parameters, including segment lengths and target buffer sizes, while adapting the network environment using variations in bandwidth and packet loss. Based on the conducted experiments, the authors propose a model to derive stalling events' relation to QoE using linear regression.

A recent analysis on the interaction of DASH with network throughput is given by [WMW17] showing that throughput can be described as a stochastic model of request durations in dynamically sized DASH segment requests for WiFi networks. The findings are verified in a real-world WiFi testbed using two DASH players,

DASH.JS and TAPAS[De +14], and thus provide a new approach for generating workloads in DASH network experiments.

A study published by Samain et al. [Sam+17] evaluates the performance of DASH AA in the contexts of TCP/IP and Information-centric Network (ICN) (see Subsection 3.3.1). Here, the authors consider three classes of AAs: BBA, TBA and a hybrid AA, each represented by the recent AA, i. e., `BOLA`, `PANDA`, and `AdapTech`, respectively. For their evaluation, each algorithm has been implemented within the `libdash` framework.

Apart from work focusing only on the underlying network mechanisms, a study with a focus on interdependencies of DASH players and the network context was conducted by Zabrovskiy et al. [Zab+17]. The presented work proposes to analyze and compare Shaka Player, DASH.JS as well as a wide range of other players focusing on QoE measures.

3.2.3 Discussion

Along with the overview provided in Table 2 and the preceding discussion of well-acclaimed AA in research and industry, it can be deduced that *many* possible approaches and solutions exist, providing objectively measured and demonstrated performance improvements—yet, a consensus on what strategy is the *best* is not trivial to deduce from such studies.

We propose the following reasons for this mismatch: First, an objective measure of when an algorithm is performing *best* is not clearly defined. While there are many accepted indicators, e. g., stalling, playback bitrate, and adaptations, a universal combined measure is itself subject to research and depends on the usage scenario (for the discussion of this see Subsection 2.4.2). Such scenarios vary widely and require the consideration of a multitude of factors by service providers with regard to the environment-context (e. g., bandwidth, delay, loss), the execution-context of the AA (i. e., DASH-player, device), and finally user expectations. It is therefore not surprising that evaluation scenarios vary widely between studies of DASH Adaptation Algorithms and focus on different objective aspects for improvement.

Last, given that many different algorithms and their extensive evaluation in the given configuration space are very time and cost intensive. For example, researchers may choose between a high level of abstraction in simulative studies or mimicking real-world usage in user-studies as an alternative. The former can quickly evaluate a large configuration space but may not adequately correspond to real-world usage and environment scenarios. The latter, however, is very costly and time-consuming and thus limits the evaluation space itself. [17]

[17] *A practice used in industry is A/B Testing, described by Netflix[ab] and has also been employed for evaluations in [Hua+14]*

A trade-off between both options are emulative approaches. While they are automated and do not require direct user interaction, they utilize the same systems for evaluation as used in a real-world setting. Yet, network properties can be centrally controlled and repeated to achieve a high level of control and confidence for comparative studies. In contrast to an event-based simulative approach, scalability

[ab] https://medium.com/netflix-techblog/a-b-testing-and-beyond-improving-the-netflix-streaming-experience-with-experimentation-and-data-5b0ae9295bdf

is harder to achieve; given that the systems execute in real-time, execution of a single DASH streaming experiment usually takes several minutes. Here, the cloud computing paradigm allows to flexibility achieve a high degree of parallelization and thus, enables to achieve large-scale experimentation while maintaining adequate runtime. In the studies presented in this dissertation, the *large-scale emulation* concept has been adopted throughout the conducted evaluations.

3.3 Video Streaming in Novel Network Architectures

We categorize our discussion on related work that applies novel networking concepts and architectures for Adaptive Video Streaming (AVS) in three areas: First, studies related to Information-centric Network (ICN), in particular concerning the Named Data Networking (NDN) specification, are presented. Here, we discuss research on DASH AA used over NDNs, including work that investigates the influence of caching as well as forwarding and mobility schemes.

Second, we present the interaction of PHY layer control for improved AVS performance in wireless networking, in particular focusing on Software-Defined Radio (SDR) concepts related to the transfer of SVC streams.

Last, we provide a very brief overview on recent developments in DASH research using Quick UDP Internet Connections (QUIC) and Multipath-TCP (MPTCP) as extensions or replacement for current TCP-based OTT video transmission.

3.3.1 NDN-based DASH

Regarding the use of ICN as an underlying network mechanism for DASH, the first publications related to this topic were experimental studies of video streaming over Content-Centric Networking (CCN) such as conducted by Awiphan et al. [Awi+13] and Liu et al. [Liu+13]. While those studies explore how varying configurations and parameters, e. g., the overlay path, the chunk size, and caching strategies affect the bandwidth usage in DASH, they are not central to the discussion in this work given our focus on NDN as opposed to the similar CCN.

Here, one of the first studies was conducted by Grandl et al. [GSW13] investigated caching in DASH over NDN. Their findings indicate that in-network caching often leads to video quality oscillations because with standard DASH a cache hit will be interpreted as an improvement in network conditions, which is consequentially followed by an adaptation to a higher video quality. This change can then result in cache misses if the next segment is not available at the cache, followed by an adaptation back to the lower video quality. Thus, the authors propose an in-network mechanism, i. e., in every cache only the segment with the highest bitrate is stored, meaning that lower bitrate segments are replaced. Also, a transcoding mechanism is attached to the cache so that every interest requesting the same or a lower bitrate can be satisfied by, if necessary, transcoding the higher bitrate to the requested bitrate.

A systematic comparison of DASH over NDN or TCP has been published by Samain et al. [Sam+17] (also see Subsection 3.2.2). The presented results indicate that the basic version of an NDN leads to reduced performance compared with TCP when used in the context of DASH. Therefore, Samain et al. [Sam+17]

propose an extension for NDNs for in-network loss recovery (i. e., wireless loss detection and recovery) that performs at the same level as TCP. With a load balancing mechanism for NDN in addition to the aforementioned in-network loss recovery, it is shown that NDN can outperform TCP in terms of measured QoE metrics in DASH. Furthermore, the authors employ a technique for a finer-grained bandwidth estimation in DASH on the NDN chunk level.

Another approach for a better interplay between DASH and NDN is the use of SVC as suggested by Rainer et al. [RPH16]. Since SVC layers are building upon each other, the caching problem in NDN becomes less harmful. All clients must request at least the base layer which increases the likeliness of cache hits. However, it is worth mentioning that in the discussed case only three different quality levels are provided which is distinctly less than in standard AVS use cases. Thus, cache misses for enhancement layers are still likely, especially when increasing the number of representations.

Another prominent area in NDN research focusses on practical challenges for DASH streaming by leveraging a JavaScript compatible version of NDN, called *ndn.js*.[ac]

Here, Muto et al. [MKK15] introduced the first implementation of a JavaScript-based NDN DASH Player to support mobile streaming; they evaluated a train commuting scenario, where video elements are prefetched using NDN to servers located in railway stations. The work was implemented on a virtualized LXC testbed. In a follow-up work by Kanai et al. [Kan+15] the system then was evaluated in a real-world scenario by deploying the proposed prefetching infrastructure in train stations and on trains. The developed DASH player was tested on mobile devices.

Last, a study by Ishizu et al. [Ish+15] proposes to improve the efficiency of NDN requests by implementing interest aggregation and evaluating the influence of different playback buffer sizes for a DASH streaming scenario. The authors implemented functionality for fetching DASH segments based on *ndn.js*, however, they did not integrate an adaptive bitrate streaming.

3.3.2 SDR-based Adaptive Video Streaming

Most work on the interaction of PHY-Layer width AVS focused on mobile networks, e. g. [Fu+13] that propose QoE aware scheduling of streams, corresponding to the SVC layers based on extensions provided in LTE. Regarding WiFi networks, an early work investigates the use of Multiple-Input and Multiple-Output aware scheduling for layered video transmission [JE08] with a focus on energy consumption. Last, work on the interaction of adaptively applying Forward Error Correction (FEC) schemes for SVC layers has been published by [NHF07].

3.3.3 New Transport Protocols for Dynamic Adaptive Streaming

While not directly studied in this dissertation, we briefly present related work to DASH streaming based on novel transport protocol concepts, namely QUIC and MPTCP.

[ac] https://github.com/named-data/ndn-js

First, given the sizeable degree of direct control on the client- and server-side network endpoints by Google[ad,18] a wide deployment of QUIC for DASH streaming in YouTube has been realized. In [Lan+17] the authors and engineers share their observations using this deployment. Most notably, they report that QUIC "reduces rebuffer rates of YouTube playbacks by 18.0% for desktop users and 15.3% for mobile users" [Lan+17]. The authors hint at several possible reasons for this performance improvement, e. g., 0 – RTT session re-initiation, multiplexing without head-of-line blocking between applications' flows, and improved RTT measurements. Other work on QUIC includes [BRZ17].

[18] *As of Wednesday 11th April, 2018 Google holds 57.69% of the browser market share with Chrome.*

The last area to be discussed as part of evolving transport protocols is MPTCP. Here, [Han+16] describe one of the first implementations of a DASH-optimized MPTCP scheduler as an extension to the Linux kernel. The authors report that, in conjunction with the Throughput-based Adaptation (TBA) `FESTIVE` by Jiang et al. [JSZ14] and Buffer-based Adaptation (BBA) `BBA` by Huang, Johari, et al. [Hua+14] (for a discussion of those AAs see Subsection 3.2.1) that mobile-network resources can be saved when applying the proposed MPTCP schedulers in varying network conditions.

A more generic approach for MPTCP, that uses the concept of a dedicated programming model, has been proposed and evaluated by Frömmgen, Rizk, et al. [Frö+17]. This openly available framework, called *ProgMP*[ae] allows the rapid specification of *efficient* schedulers, executed in kernel space, that can be applied for use cases such as DASH, as discussed by the authors.

3.3.4 Discussion

Given this brief introduction of related work on novel network architectures and transport protocols used in DASH streaming two, general observations can be made: first, a wide variety of concepts are currently being explored simultaneously, e. g., NDN, SDR, QUIC, and MPTCP. Secondly, the evolvement and time to market are very rapid, as seen in the example of QUIC that has been widely deployed before its final specification by the IETF. Therefore, the focus in this work lies in more long-term developments in the Internet by investigating practical challenges for DASH in NDN. While most work on NDN has been conducted using simulative approaches (e. g., [Bha+15; JSC17; Pos+14; Rot+17; RPH16]) or in architectures with limited practical topology size (e. g., [Sam+17]) one goal in this work is to explore performance of DASH in NDN and address limitations of current Adaptation Algorithms for NDN. To support these evaluations, we predominantly use two evaluation frameworks. First, a container-based testbed in emulation environments, and second, a real-world wireless testbed.

[ad] http://gs.statcounter.com/browser-market-share
[ae] https://progmp.net/

4 Mobile User-Generated Video Composition and Distribution

The first contribution presented in this dissertation is a novel approach for the real-time composition of User-Generated Video (UGV) streams in mobile and dynamic scenarios.

To begin with, an analysis of user behavior in Mobile Video Broadcasting Services (MBSs) is presented, with a detailed investigation of further aspects, such as video quality for the largest platform at the time–YouNow. In this study, we show critical aspects in the delivery component of User-Generated Video composition systems with regard to mobile video up- and down-streaming; for example, as a result of this study, we see a direct relationship between a reduction in the quality of video streams and their origin from mobile devices and networks. This motivates research of live-User-Generated Video in challenging and dynamic conditions, common in mobile networks and large crowds. Similarly, relevant work shows that the extensive coverage of events by creators of live-User-Generated Video poses a challenge for viewers [Wil16]; identifying and switching between relevant, high-quality views is a non-trivial task, especially with an increase in the supply of live views to select from.

We deduce that a crucial step for maintaining a high user satisfaction in the composition and delivery of live-User-Generated Video is to identify at the source (i. e., on the uploading device), which views are relevant and allow to achieve a high resulting Quality of Experience (QoE) when selected in compositions. Here, our contribution is to develop efficient mechanisms that can automatically select such sources with a low overhead in terms of device and network resources. As discussed in Subsection 3.1.1, this has not yet been addressed by research, as the focus has been on a content-centric analysis of video streams which induces a high overhead on networks and mobile devices.

Thus, based on the derived technical requirements and challenges for live UGV, we propose an approach for the efficient composition of multiple live User-Generated Video sequences, taking into account video quality, and network and device context information. A prototype of this approach was implemented and evaluated using a crowd-sourced user study showing that this concept allows for a *low-overhead* and *low-delay* composition of mobile video and improved QoE. Further, we experimentally validate the real-time applicability as a Quality of Service (QoS) attribute of this approach.

This chapter describes ideas, concepts, and results presented in our peer-reviewed publications [Sto+15a; Sto+16c; Sto+17b; SWE14]

4.1 *Analysis of YouNow*

The following section uses YouNow as a representative example of technical analysis of an MBS. To motivate this choice, we begin with an overview of YouNow, compare it to other platforms, and continue with details on YouNow's architecture. Based on this, we then present aspects of user behavior along the lines of workload, content popularity, and system characteristics.

4.2 *YouNow vs. other Mobile Video Broadcasting Services*

This section highlights key characteristics of MBSs based on data that has been sampled from the platforms Bambuser[af], uStream[ag], Meerkat[ah] and YouNow[ai] between Friday 26th June, 2015, and Thursday 2nd July, 2015. For the analysis of the systems, we periodically (ca. every 5 minutes) requested data from the APIs, in the same way as they would have been used when visiting the website. This primary analysis is followed by a more detailed investigation of YouNow based on operational data sampled between the Sunday 15th March, 2015 and Thursday 9th April, 2015, presented in the following sections.

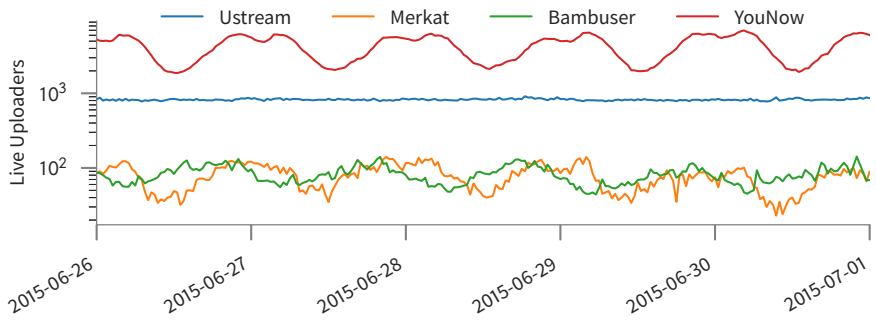
In Figure 13 we show the popularity patterns from two perspectives: *i)* the broadcasting side *ii)* and the viewer side.

On average, the discussed platforms have between 42 and 2687 live broadcasts, where a broadcaster attracted around 9 viewers for YouNow, 20 for Meerkat, 20 for uStream, and 71 for Bambuser. The platforms Meerkat and Bambuser had fewest broadcasters with less than 75 in the time measured. However, their content remains popular, with up to 18 000 concurrent views for Bambuser and 27 000 for uStream. Meerkat, which has ceased its operations on Friday 30th September, 2016, has been attracting a relatively low number of uploads (up to 180 live video streams), achieving between 194 and 5996 viewers. YouNow is the most promising platform, as it attracted most viewers and recorders in 2015, with a peak number of 6971 broadcasters and 90 599 viewers watching video streams in parallel at peak times.

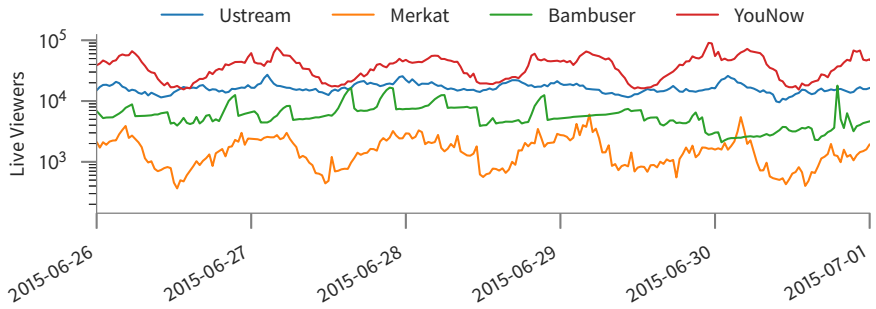
Regardless of the individual patterns of video uploads, all streaming platforms achieved high upload to viewing ratios in comparison to Video Streaming System (VSS) such as YouTube [Cha+07]. This is because MBSs broadcasters only compete for viewers of other live streams and not the full history of prerecorded video clips that is significantly larger. As a result, the chance for a new user to attract viewers is significantly higher.

As shown in Figure 13, YouNow was, at the time when the analysis was conducted, the most successful MBS among the investigated platforms. We acknowledge that today the market in MBS has changed significantly, with companies such as YouTube and Facebook now also providing live User-Generated Video services.

[af] bambuser.com
[ag] ustream.tv
[ah] meerkatapp.co
[ai] younow.com



(a) Concurrent Broadcasters



(b) Concurrent Viewers

Figure 13: Overview numbers of available streams and viewers of four popular MBS platforms: Bambuser, Meerkat, uStream and YouNow. *a)* The y-axis depicts the number of concurrent broadcasters; *b)* the y-axis represents the number of concurrent viewers.

4.2.1 YouNow Architecture

The main source of the videos available on the YouNow platform is the upload from mobile devices such as smartphones and tablets. Thus, YouNow provides dedicated streaming applications for the mobile platforms Android[aj] and iOS[ak]. The main task of those applications is to allow users to stream their content and watch streams provided by other users.

These functionalities can also be accessed on YouNows's webpage. This website uses a Representational State Transfer (REST) API providing JavaScript Object Notation formatted responses that build the data basis for the web user interface. An overview of the existing REST resources is listed in Table 3. When viewers or broadcasters are visiting the website, requests to those Uniform Resource Locators are sent and the returned data and is used for rendering the page with the JavaScript-based Mobile Video Composition (MVC) framework AngularJS[al]. For video playback, a flash container is generated and embedded in the website

[aj] https://play.google.com/store/apps/details?id=younow.live
[ak] https://itunes.apple.com/us/app/younow-broadcast-chat-watch/id471347413
[al] https://angularjs.org/

URL	Name	Description
http://www.younow.com/php/api/younow/trendingUsers/numberOfRecords=10	trendingUsers	Lists broadcasters ranked by their trending status
http://cdn2.younow.com/php/api/broadcast/playData/channelId=<channelId>	playData	Data for generating the RTMP playback URL given a channelId
http://www.younow.com/php/api/broadcast/info/user=<username>	info	Detailed user information given a username
http://cdn2.younow.com/php/api/channel/onlineUsers/channelId=<channelId>	onlineUsers	Data about viewers currently watching a broadcast session

Table 3: URLs and description of the YouNow REST API used for data collection.

that allows video playback using RTMP. The video streams are encoded using H.264/AVC and delivered using the *Wowza Streaming Engine*[am] via the Akamai Content Delivery Network (CDN).

Apart from data relevant for rendering the website and presenting information directly depicted to viewers, internal reporting data is included in the requests' responses. It has been used for analysis in this paper. An overview of all fields available for analysis is given in Table 4. This data contains information regarding the internal reporting system of broadcasting users such as device and connection types of broadcasters, additional user profile information, and stream quality indicators.

4.2.2 Broadcasting and Viewing Workload

The workload of an MBS, such as YouNow, is determined by multiple factors, as pointed out in [ZL15]. The number of concurrent broadcasters and viewers is one such factor. This property plays a key role in characterizing the usage patterns of the YouNow service to improve its scalability. To understand the dynamic behaviors of the number of users of the YouNow video service, including broadcasters and viewers, we count the total number of unique broadcasters for every hour within one week, from Wednesday 6th May, 2015 to Tuesday 12th May, 2015. Figure 14 shows the daily access patterns of the number of broadcasters from all countries, the US and the UK.

Figure 14 shows the daily workload patterns regarding the number of broadcasters from all countries, as well as the share of broadcasters from the US and the UK. These counties constitute the majority of broadcasters in YouNow. It shows that broadcasters tend to upload videos mostly in the evening and the number of broadcasters peaks around mid-night of GMT. Broadcasters gradually leave the YouNow video service in the early mornings and join broadcasting sessions in the afternoons. It is interesting to see that the peak number of broadcasters around midnight of Friday 8th May, 2015 and Saturday 9th May, 2015 is higher than on other days. The lowest number of broadcasters on Saturday 9th May, 2015 and Sunday 10th May, 2015 are also higher than on other days possibly due to being weekend days.

[am] http://www.wowza.com/streaming/live-video-streaming

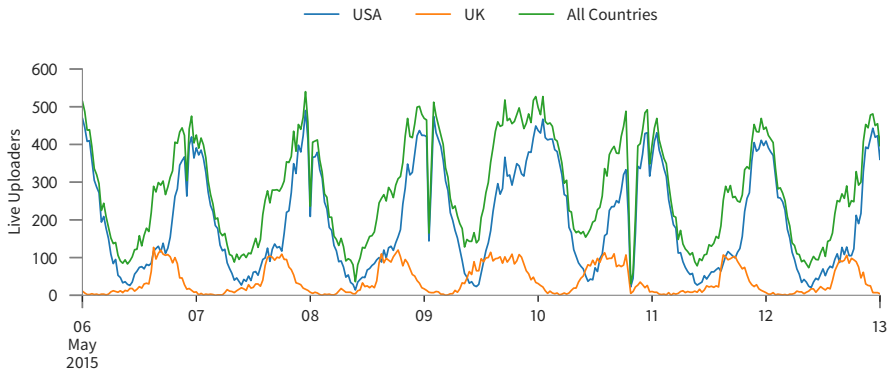


Figure 14: Plot of the total number of broadcasters on the YouNow platform for the period between May 06 to May 13, 2015.

Different time zones cause multiple peaks in the number of broadcasters within one day. However, the largest peaks can be mainly accounted to the 70% of broadcasters from the US, as depicted in Figure 14. Considering the time differences between US and GMT timezones, the number of broadcasters from the US reaches the peak in the late afternoon of their local time. Similarly, by examining the

Name	Fields
trendingUsers	`userId, viewers, likes, tags, broadcastId, username, userlevel, profile, locale, shares, fans, totalFans, lastPosition, position, total`
playData	`serverTime, channelId, copy, length, shares, quality (bitrate, fps, kfr, percent, desc, high) dynamicPricedGoodies (PROPOSAL_RING, 50_LIKES, FANMAIL, CHATCOOLDOWN), stickersMultiplier, queues, positions, broadcastId, nextRefresh, nextRefreshMobile`
info	`userId, youtubeStart, giftsValue, lastTopFanAnnounceNew, display_viewers, lastBelowVideoGift, broadcasterBoostLevel, state, media, topFansCount, lastMonitorCheck, dateCreated, coins, mirror, friendsReq, referrals, origCountry, mviewers, username, partner, broadcastId, points, maxTUScore, locale, stateCopy, topFanNew, userlevel, minChatLevel, title, platform, origSettings (bitrate, fps, kf, tcp, videoSize), location, quality, geoLocale, likes, maxConcurrentViewers, brScore, barsEarned, reconnects, stickersMultiplier, shares, totalFans, followersStart, monitorDisconnect, vip, settingsId, dateMonitorDisconnect, lastQuality (bitrate, fps, kfr, percent, desc), facebookId, facebookOption, facebookUrl, twitterHandle, googleHandle, userLevel, description, firstName, lastName, totalFans, youTubeUserName, youTubeChannelId, youTubeTitle, viewers, broadcasterInfo, featuredTime, acceptLanguage, qualitySamples, fbPublish, country, dateStarted, profile, language, broadcastsCount, premiere, maxLikesInBroadcast, twPublish, likePercent, serverTime, length, comments`
onlineUsers	`users, nextRefresh, totalUsers`

Table 4: Data fields provided by the YouNow REST API.

access patterns of broadcasters from the UK, there is a peak for video uploads from 3:00 PM to the evening hours of their local time. Similar to the analysis

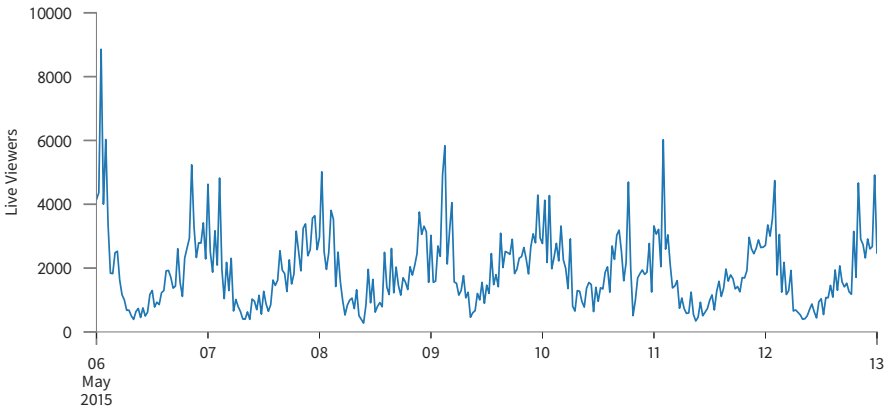


Figure 15: Plot of the total number of viewers on the YouNow platform for the period between May 6 to May 13, 2015.

performed on the access pattern of broadcasters, we also show an aggregated view of the number of concurrent online viewers for the same period in Figure 15. The access pattern of viewers are also strongly time-dependent. It peaks to over 8000 around mid-nights and falls back below 1000 at noon. Thus, we can conclude that the viewers of YouNow also tend to watch more videos in the evenings (GMT). Last, the geographical distribution of unique broadcasters worldwide is depicted

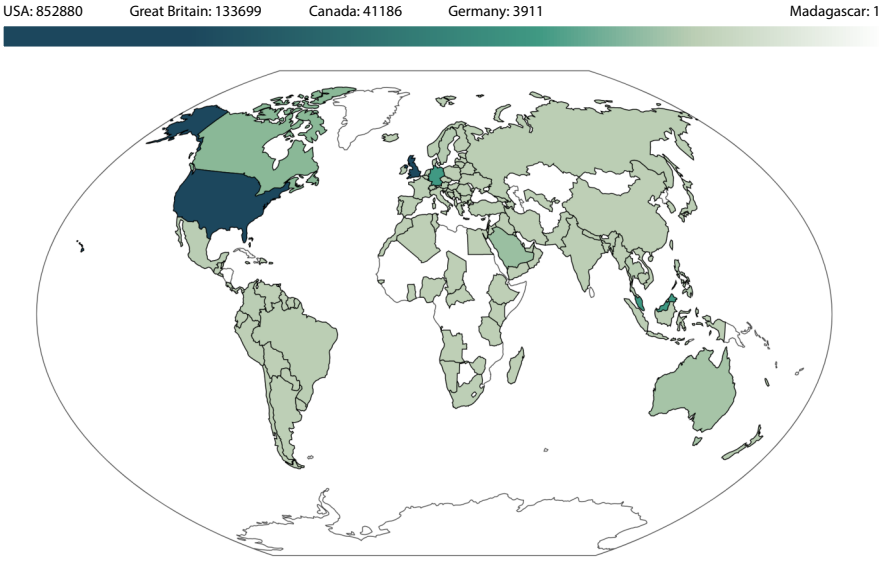


Figure 16: Number of unique broadcasters per country.

in Figure 16. The USA has the clear dominance in terms of broadcasters, followed by the European countries UK and Germany.

4.2.3 Session Duration and Online Time of Broadcasters

A broadcasting session is defined as the duration a user remains in a live stream. For all logged sessions, we show the Empirical Cumulative Density Function (ECDF) of these durations in Figure 17.

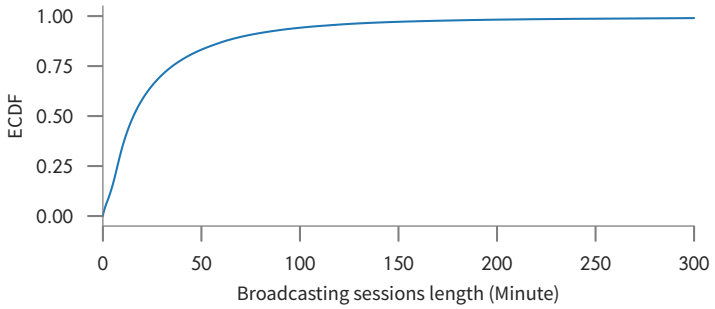


Figure 17: Empirical Cumulative Density Function of session duration and the total online time of broadcasters.

Like the analysis performed on the number of broadcasters and viewers in Subsection 4.2.2, this analysis is based on the data collected from May 06 to May 12, 2015.

About 93% of the broadcasting sessions last less than 100 minutes, and only 14% of the broadcasting sessions last longer than one hour [ZL15]. The duration of broadcasting sessions on YouNow are generally shorter than those on other similar broadcasting services, like Twitch.tv, where 30% of the sessions last more than four hours. A possible reason for this is that a large share of broadcasting sessions originate from mobile devices and mobile networks (see Subsection 4.2.5, Subsection 4.2.6), which imposes limits on the duration in terms of data contracts and battery time compared to streaming from stationary clients.

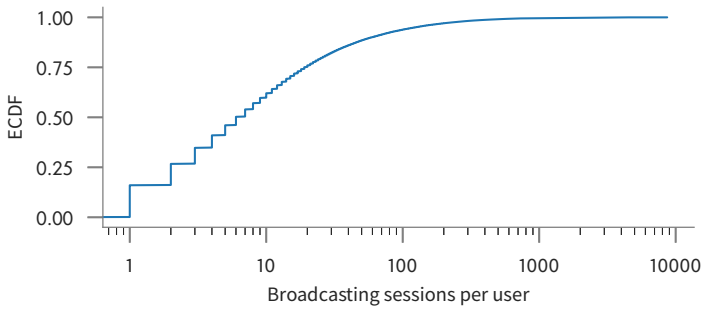


Figure 18: CDF of the number of broadcasting sessions initiated by users.

The distribution of the number of sessions initiated by broadcasters within one week is plotted in Figure 18. This figure shows that over 20% of the broadcasters initiated only one broadcasting session and about 25% of the broadcasters created more than ten broadcasting sessions within the considered week. This

demonstrates that there exists a group of *highly active* broadcasters that is willing to broadcast several times daily.

4.2.4 *Broadcaster Popularity*

The popularity of broadcasters is an important factor for viewers, broadcasters and the YouNow platform itself. Viewers may want to know about most popular broadcasters to follow interesting personal shows, while YouNow promotes revenue models and shares profits with popular broadcasters attracting a large amount of viewers[an]

Based on the information of viewers in relation to each broadcaster, we have analyzed sessions of 85994 broadcasters to investigate the relationship between the overall number of viewers and the platform's top broadcasts. Here the top 10% of broadcasts are responsible for more than 80% of all views. On the other hand, as depicted in Figure 19, more than 5% percent of broadcasters do not attract any viewers.

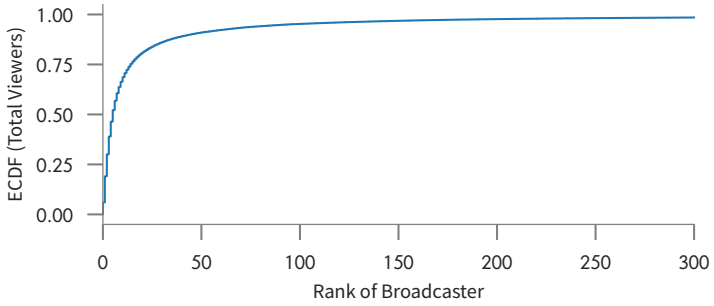


Figure 19: Relation between the share of total views and top-ranked viewers.

This shows that most of the platform load can indeed be accounted to a small fraction of broadcasters. At the same time, a large share of the bandwidth used for receiving streams is spent without gaining any additional viewers and thus revenue.

4.2.5 *Device Usage*

Given the data about the device types used by broadcasters, we have ranked all devices according to their frequency as presented in Table 5.

As shown here, there is a clear dominance of Apple devices for video broadcasting, having a share of almost 70% of all devices used. Following that, there is a large diversity of devices using the Android system.

The US market share of Apple and Android smartphones was 54% and 45% respectively, in 2015.[ao] Therefore, there is a clear shift towards the dominance of Apple devices used to broadcast video streams in the YouNow platform in contrast to the distribution of the US market.

[an] http://www.younow.com/partners

[ao] http://gs.statcounter.com/os-market-share/mobile/united-states-of-america/2015

Device type	% of devices
iPhone5	14.38
iPhone6	10.56
iPhone7	9.79
iPad2	9.78
iPod5	7.00
iPhone4	5.91
iPad4	4.01
iPhone3	3.59
iPad3	2.94
iPad5	0.65
LGE LG-D415	0.62
samsung SM-T230NU	0.58
LGE LGMS323	0.56
samsung SCH-I545	0.54
samsung SM-G386T	0.42
Others	28.24

Table 5: Percentage of reported device types for each broadcast session as reported by the YouNow API.

Connection type	% of connections
4G	3.0458
3G	1.4307
2G	0.0035
WiFi	29.8915
Undefined	65.6264

Table 6: Percentage of occurrences of broadcast session connection data for different connection types as reported by the YouNow API.

4.2.6 Broadcasting Quality

Given the data included in YouNow's internal streaming quality reporting system, we have analyzed the relation between video encoding parameters and the network used for broadcasting streams. The data classifies different network types, which allow distinguishing between uploads originating from mobile provider networks and WiFi. Further, there are three types of unclassifiable data (entries tagged *Unknown* and without any value which we have omitted given their respective small size of instances) and *Undefined*.

The data is plotted in Figure 20 for the video bitrate and Figure 21 for the Frames per Second (FPS). The video bitrate is dependent on the type of network used for the upload. Here, when comparing 2G and 3G/4G networks, the uploading bitrate for the latter is higher in almost all cases. WiFi connections show a high bitrate variance, with values in the 25th and 75th percentiles overlapping with bitrates of 3G/4G connections.

A possible explanation for this is that the access networks for the WiFi connections are very diverse and cause large differences in up- and download bandwidths for this connection type reported by the system. Based on the observable ranges of bitrates for different network connection types, there is evidence for the existence of an adaptive broadcast upload determining the bitrate to be used, either based on the network type or bandwidth measurements.

Also, for the *Undefined* connection category, given the shape of the kernel density estimate depicted for both bitrate and FPS figures, two clearly separated clusters can be identified in each. Thus, it can be reasoned that this category represents two distinct connection types. However, we can not further judge on the connection types represented by these clusters.

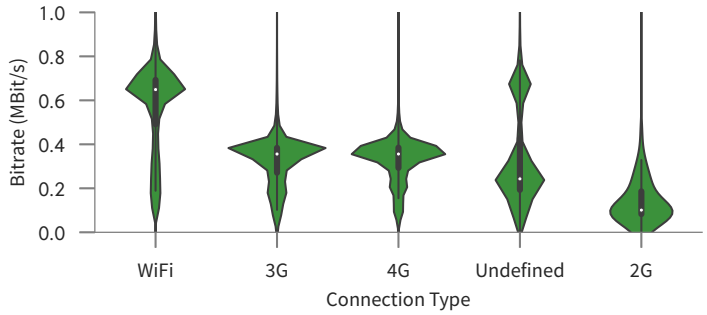


Figure 20: Violinplots of the uploaded videos' bitrates for different connection types showing the median and distribution by connection type. Inner black boxes represent median and IQR, with extending lines indicating the full distribution span. Further, the green areas indicate this distribution as a kernel density estimate, with the respective width representing the relative number of observations.

Looking at the FPS of the video stream shown in Figure 21, there is also significant difference between 2G and other connections. However, for most cases, the FPS does not reach a value higher than 15, independent of the connection type.

Analysis Summary

In this section, we investigated the encoding parameters of video in MBSs and identified that bitrates, frame rates and resolutions of video streams generated for live mobile upload exhibit limited quality. Overall, bitrates are low compared to Video on Demand (VoD) providers with up to 1 Mbps, and frame rates are below 15 frames per second. This, along with mostly short sessions durations, negatively impacts QoE, which qualifies the need to improve upload and selection in Mobile Video Composition, that fundamentally depend on such uploads to efficiently and swiftly identify relevant streams to achieve an ongoing, high QoE stream for viewers.

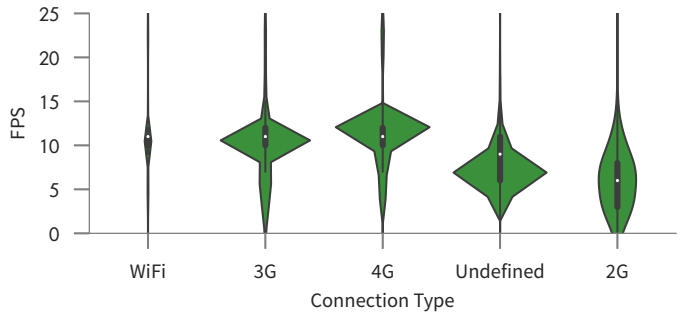


Figure 21: Violinplots of the uploaded videos' FPS showing the distribution by connection type. Inner black boxes represent median and IQR, with extending lines indicating the full distribution span. Further, the green areas indicate this distribution as a kernel density estimate, with the respective width representing the relative number of observations.

4.3 Context-Based Mobile Video Composition Support

The key goal of context-based Mobile Video Composition is to provide a method for the efficient measurement and collection of client and network data to aid the selection, upload, and composition of live User-Generated Video streams for MVC systems. An overview of the proposed process is depicted in Figure 22. Ultimately, the motivation for aiding the decision processes of MVC, as proposed here, is to enhance QoE for viewers of collaborative live streams and to eliminate the upload of unused streams—thus reducing costs for users uploading video.

We begin with a discussion of our approach with the categorization of indicators and respective metrics that constitute one of the core novelties of our approach by including the devices' sensor information as a basis for the view selection. Next, we describe how a combined indicator for a later composition, called *Recording Score* is generated. This is followed by details on the overall architecture to support the collection of metrics.

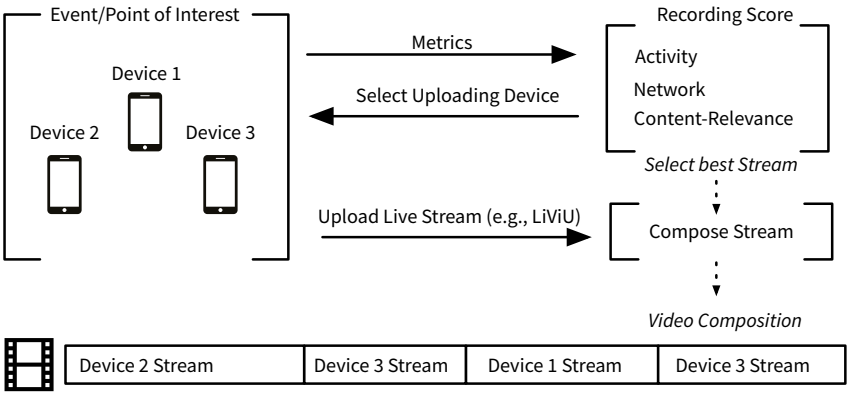


Figure 22: Architectural overview of live MVC support approach based on device metric collection and derived recording scores.

4.3.1 Context-Based Metrics and Data Collection

On a high level, the proposed method relies on the collection of metrics in four categories as a basis for deriving a combined indicator for the expected QoE of recorded streams. These are: *i)* static parameters such as camera properties collected and transmitted once, at the time when a session is initiated; *ii)* activity-related dynamic parameters, including shake estimation and activity recognition based on device sensor information; *iii)* network properties measured and compared with an estimation of the upload bandwidth; *iv)* measures on content relevance for a given Point of Interest (PoI) based on the recording location. In the following, we first describe underlying metrics used to derive a *Recording Score* indicator detailed in the next section.

Metrics

First, for location retrieval, the client uses Android's API Client to obtain the exact location. As usual, the localization is based on the GPS sensor or on visible WiFi and mobile access points. The shake detection algorithm is implemented using data from the linear accelerometer of the device. Values from the sensor are recorded along with a timestamp and are periodically sent to a shake detector which estimates whether there is shaking based on a predefined threshold for the allowed deviation values. If shaking is detected, its amplitude, velocity, and duration are calculated. Information about the current user behavior, which potentially impacts the quality of the recording, is based on the activity detection implemented using Google Play Services.[ap] The detected attributes can be found in Table 7. For the network connection, the first parameter that is recorded is the type of network the device uses. Devices connected to WiFi or LTE networks are preferred over those using a slower connection, assuming that these networks provide a superior streaming quality (see Table 7). Last, the actual available bandwidth for the upload of live-video is estimated using active probing (this process will be further detailed in the following subsections). Retrieving network status information using this estimation technique induces a far less mobile data usage than transferring the video stream itself. The results are used to calculate which devices available for uploading their streams fulfill the required bandwidth conditions. In this way switching to devices that do not allow continuous live upload is avoided.

Recording Score

Based on the above-described metrics, we continuously derive an overall recording score that builds the basis for the later explained live-composition. The parameters can be factored with different coefficients. They have been initially set to represent estimates of their relative importance based on observations when developing the discussed approach. An overview of the parameters and their weighting factors is given in Table 7.

For many multi-stream events, the distance between the recording device and the PoI has a major influence on the QoE for the viewers; examples include concerts

[ap] https://developers.google.com/android/guides/overview

Table 7: Recording score parameters

Indicator	Weight	Attributes	Value
Activity	0.5	Still, Walking, Tilting, In vehicle, *other*	5, 4, 3, 2, 1
Shaking	0.5	Yes/No	1, 5
Distance	1	x = Meters	$f(x) = -0.05x + 3.7$
Network	1	x = Bandwidth y = Video Bitrate	$f(x, y) = x/y * 5$
Network	1	WiFi, LTE, HSPAP, 3G, UMTS	5, 5, 3, 2, 1

or political demonstrations. The estimator accepts a location (PoI) and a required bitrate as initialization parameters. They are used to assigning scores for all devices currently connected to the system. For the calculation of the *device location score*, the distance between the device (based on its GPS location) and the defined PoI is derived. In case this distance is between 5 and 50 m, the linear formula for the event venue is used, as specified in Table 7 [WE14b]. For larger and smaller distances, 1 and 5 are assigned as scores, respectively.

For the *device activity score*, we consider the results from the shake estimation and activity detection. If shaking is detected, the minimal score of 1 is taken; otherwise, the maximal score of 5 is used. The activity detection algorithm assigns a maximal score to *Still* devices, and a minimal score when the detected activity is *Running*, *On bicycle* or *In Vehicle*. For *Walking* a value of 3 is assigned and *On foot* a value of 4. The average of the two indicators is the overall *device activity score*.

Next, the *device network score* measures the current network's capabilities by assessing two possible scenarios. It uses the bitrate of the video recording and assigns a maximal score to devices that have an available bandwidth equal or higher to the required one. For lower values, the score is linearly scaled between 1 and 5. In case a recent bandwidth estimate cannot be derived (e.g., due to high channel fluctuation), or the last estimate is older than 60 seconds, the score is based on the network context. If the devices are connected by WiFi or LTE, a score of 5 is assigned, while lower scores are assigned to slower network technologies. The sum of the weighted parameters is divided by the sum of weights, resulting in a score in the range of 1 to 5.

Bandwidth Estimation

As the framework only describes the interface for estimating the available upload bandwidth, various estimation approaches can be implemented. At this point, we provide two estimation techniques that have the goal to provide a high level of estimation fidelity while keeping the data overhead low.

First, as a baseline, a naive implementation was built. It first records the time on the client before starting the sending of packets, sends a number of maximum-sized User Datagram Protocol (UDP) packets (45 by default), and checks the reception time on the server. As there might be a clock skew between the client and the server, Network Time Protocol is used for determining the difference between the clocks and calculating the real transmission time of the packets. In

the end, the number of transmitted bytes is divided by the time the transmission took in order to calculate the average bitrate.

In addition to this naive approach, a more advanced approach was also implemented. The `WBest` algorithm proposed by Li, Claypool, et al. [LCK08] was selected, as it provides fast and low-overhead estimates in a mobile context. `WBest` first sends pairs of packets to determine the effective capacity C_e. The packet-pair technique for estimating the width of the narrowest link in a path is usually credited to Jacobson [Jac88] and Keshav [Kes95], it was further analyzed by Hu et al. [HS02]. It examines the behavior of the network when two packets are sent back-to-back. The exploited feature is that when the link is wide enough, the space between the two packets is preserved, whereas passing through a narrow link causes a gap between the two packets. This packet dispersion can be used to estimate the width of the narrowest link, and with it, the effective capacity C_e of the entire path. Next, several trains of packets are sent at the rate of the estimated effective capacity. The concept of sending and receiving larger sequences of packets is usually attributed to Jain et al. [JR86]. It can be used to estimate the achievable throughput on a path. `WBest` uses this feature to calculate the dispersion of the packets in each train and taking the mean as an estimate. The achievable throughput R is then calculated based on the number of bytes in a packet divided by the packet dispersion. Using this data on the realized throughput, i. e., the concrete measure of the achieved data rate, the achievable bandwidth A is then computed as $A = C_e(2 - C_e/R)$. Here, the available bandwidth describes the expected measure of realizable throughput. For a detailed description of the algorithm, we refer to the authors' original publication [LCK08].

4.3.2 Protocol and Architecture

We will now introduce the system's architecture and components, beginning with the employed messaging protocol.

For messaging, an asynchronous client-server pattern[aq] was implemented using `JeroMQ`. The process of establishing a monitoring session, as shown in Figure 54, is explained in the following. First, the client sends a `JoinInfo` message to the server, which includes static information about the device, such as the model and camera characteristics (megapixels, image stabilization capabilities, etc.), along with dynamic parameters, including the current location. The server replies with a `JoinInfoAck` message, which acknowledges the receipt of the `JoinInfo` message. It also specifies an update interval at which the device will periodically send information about the QoS parameters. After that, the client prepares and sends such messages, which include position information, shake-related data, presumed activity that the holder of the device is currently performing, and network information. The server may also request an immediate update if up-to-date data is required. At the end of a video streaming session, the client can gracefully quit by sending a `LeaveInfo` message.

In addition to this messaging protocol, the server checks the time of arrival of `UpdateInfo` messages from each active device. If the maximum time between

[aq] http://zguide.zeromq.org/page:all/#toc76

update intervals is exceeded, the server assumes that the node has gone offline, and removes it as a potential stream provider.

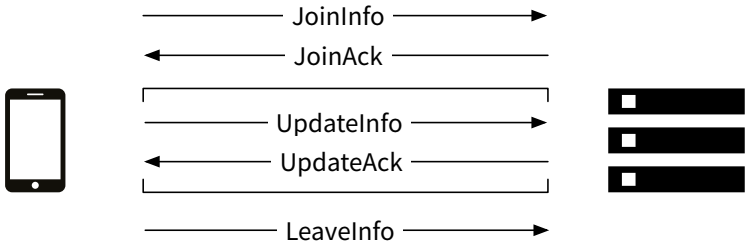


Figure 23: Messaging protocol for the monitoring framework

Data Storage

The relevant data about the expected video quality and the devices chosen for upload are saved in a `PostgreSQL` database to allow later analysis.[ar] First, at the beginning of a session, the factors of all measured parameters (shake, network information, etc.) are recorded. Then, all `JoinInfo` and `UpdateInfo` are saved as they arrive. Both, the `JoinInfo` and `UpdateInfo` messages, contain the device ID provided by the manufacturer, which can be used to retrieve information about a particular device of interest.

The estimated bandwidth for every device, as well as the amount of additional network overhead created by the bandwidth estimation, is also recorded. In the end, the usefulness score of all devices and the device picked for their upload are also saved.

Integration of the Monitoring Framework

The client-side functionality is packaged as an Android service that can be started and stopped via an Intent. The information collected and analyzed by the monitoring framework is exposed by the server via methods, as well as through a REST API. Thus, the framework can easily be used by other composition systems. The information that can be retrieved includes the latest `JoinInfo` and `UpdateInfo` messages from the connected devices. Furthermore, the recording score of these devices and the best upload device selection are also stored.

4.3.3 *The Video Recording Application: LiViU*

The Live Video Upload System (LiViU) [WKE15] is an *adaptive mobile video upload system* that can be used in conjunction with TCP or UDP. It is quality-adaptive as it supports the transcoding of multiple video representations in parallel on Android mobile devices. Also, it enables adaptations in the scheduling mechanisms in order to allow a video composition system to request video chunks. Both concepts are depicted in Figure 24.

[ar] http://www.hsqldb.org/

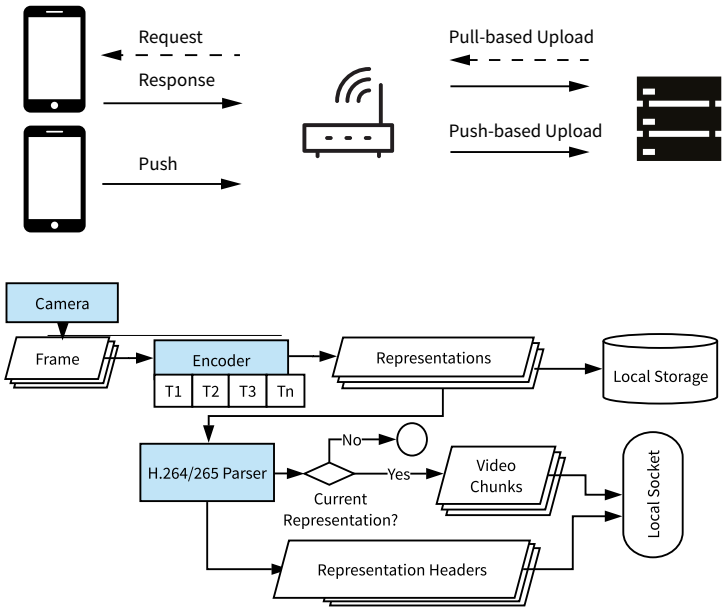


Figure 24: Overview on the concepts and mechanisms of LiViU. The upper part of the figure shows the two streaming mechanisms supported by LiViU: push and pull-based upload. The lower part of the figure depicts the encoding process on Android that supports the live generation of adaptive quality stream layers.

Adaptive Video Upload

LiViU allows the creation of multiple video representations in real-time. By generating different bitrate versions of a video, it can ensure that the available bandwidth for an upload is utilized optimally. LiViU extends the media recording Application Programming Interface on Android phones to set up different encoding threads. The hardware encoding available on current smartphones can be leveraged for transcoding videos. Each raw video frame retrieved by the camera is handed over to the video encoding thread. The graphics rendering API of Android is used to run the transcoding on the GPU of the device. The raw frame is converted into a two-dimensional texture, which is represented as a three-dimensional texture if a set of frames is available. To access the GPU, the OpenGLES library for mobile devices is used[as], which allows a quick manipulation of the resolution and frame rate. Each encoding thread operates on a copy of the texture and manipulates it according to the desired frame rate and resolution. The final step hands the texture buffer over to the respective video encoding unit which leverages the built-in hardware to encode the representation at the desired bitrate. The resulting H.264/AVC video representations are consecutively written to the device's main memory.

A real-time capable video parsing service analyzes the consecutively written video frames and offers them to LiViU for transmission. The complexity of understanding when a switch can be conducted without visual impairment (i. e., artifacts due to switches in encoding) on the receiver side is hidden within the parsing

[as] https://source.android.com/devices/graphics/arch-egl-opengl.html

service. Video chunks of the selected representation are handed over to the LiViU transmission using a local socket on the mobile device.

Adaptive Scheduling

LiViU can adapt between two different scheduling schemes: push-based and pull-based delivery (see Figure 24). In this work, LiViU uses a push-based delivery to allow a low-delay streaming with minimal overhead. Yet, a switch between the two schemes can be triggered by the server. The pull-based delivery of chunks is controlled by the application on the receiver side, which can determine when to request which media chunk.

4.3.4 Composition

Video composition can be performed as an independent component of the application. In this work, we employ two approaches for composition: offline and online.

LiViU saves the generated streams with a given `deviceID` and `sessionID` as files on the server. The information from the monitoring system is saved in a database and can thus also be used for offline processing. The saved information includes all messages exchanged by the devices and the server, as well as the time of their sending. Thus, all the parameters relevant to the video quality can be extracted and analyzed. For the offline composition, a Python-based system using the *moviePy* library[at] is employed, as detailed below in the Section 4.4.

In addition, an online composition system was implemented in Java, using OpenCV[au]. It generates and displays the resulting composition with a very low overhead in real-time. It was used for the system's evaluation in Subsection 4.4.3.

Formally, the process of video composition of video streams recorded in-parallel S_{V_I} can be described as follows. The composed output video consists of a set of video shots S_C, where each video shot represents an uninterrupted sequence of frames from one video stream of V_I.

In line with the automatic composition as proposed by, e. g., Shrestha et al. [Shr+10] that includes a filtering step and a composition step, we adapt this filtering by selection of uploading devices. Thus, we pre-select video streams which have a certain minimum video quality Q_{min}.

The final decision which input video stream should be shown in the composed video stream is then based on the $Q(V_I)$, i. e., the *Recording Score*, and a diversity score which ensures that all input videos have a chance to be part of the final composition at some point in time. This diversity score allows, in our proposed approach, to assign additional requirements for the video quality assessment such as a penalty for recently selected input video streams. This is to minimize their likeliness to be selected for composition in the upcoming composition decisions—thus increasing shot diversity in the final composed video output S_C.

[at] http://zulko.github.io/moviepy/
[au] http://docs.opencv.org/

4.4 Evaluation

The proposed approach and the corresponding system were evaluated with the goal to compare the performance of video composition using the context-based MVC approach to unsupported, i. e., naive video composition.

To this end, in a field test, we recorded videos at five locations with five mobile devices and five users. The recordings were stored for offline processing on both, the device and the server. The latter case, i. e., live transmission of video, therefore captures all potential effects by the network when transmitting video from mobile devices using LiViU. For each of the recording locations, two composed videos were created, one using the recording score $Q(V_I)$ and diversity requirements as the basis for composition, and one with a random selection of shots instead of a recording score based selection, yet, maintaining diversity requirements between the five streams.

In a second step, we then employed user studies to derive user satisfaction (i. e., Mean Opinion Score (MOS) and Just Noticeable Difference (JND)) for the composed video sequences. Last, we analyzed the live streaming capabilities of our system in an online scenario.

4.4.1 Evaluation Setup

As mentioned, in the first part of our evaluation we employed a field study for the creation of five video sequences, stitching the transmitted sequences together offline. The video sequences were simultaneously recorded with five Nexus 5 smartphones in five nearby locations in a park, each carried by an individual user. The PoI was determined at the beginning of each session, as depicted in Figure 25[av] by arrows.

Each device was operated by the recording user for a duration of at least three minutes. Monitoring data was collected by the prototype implementation of our the context-based composition system and transmitted to the server to be stored in a `PostgreSQL` database, along with the streamed video sequences. Both services were hosted on an `Ubuntu 16.04` server connected to the university network with a bandwidth of 1 Gbps. The network connections of the recording devices were configured so that a wide range of connection attributes was achieved, as follows: two devices were using a WiFi network with a mobile access point, connected via LTE to the Internet with a maximum upload bandwidth of 50 Mbps. Two other devices used LTE directly with up to 7.2 Mbps and 50 Mbps bandwidth, respectively. The last device used a 3G connection with up to 7.2 Mbps. Each device maintained the network settings while the user instructions were rotated for each location. An overview of the setup is given in Table 8.

We defined the streaming quality in LiViU to have a bitrate of 3 Mbps with 30 FPS, a video resolution of 1280×720 pixels, encoded with H.264. During recording, LiViU adapted the bitrate of the recording dynamically between 3 Mbps and 1.5 Mbps based on bandwidth measurements provided by the monitoring framework.

[av] http://maps.stamen.com/toner/#16/49.8769/8.6539

Location	Device (Network)	Movement	Shaking	Panning
		(o: none; +: some; ++: intensive)		
Location 1	1 (LTE 7.2 Mbps)	o	o	o
	2 (3G 7.2 Mbps)	+	++	o
	3 (WiFi 50 Mbps)	+	+	o
	4 (WiFi 50 Mbps)	o	o	+
	5 (LTE 50 Mbps)	+	o	o
Location 2	1 (LTE 7.2 Mbps)	+	+	o
	2 (3G 7.2 Mbps)	o	o	+
	3 (WiFi 50 Mbps)	+	o	o
	4 (WiFi 50 Mbps)	o	o	o
	5 (LTE 50 Mbps)	+	++	o
Location 3	1 (LTE 7.2 Mbps)	+	o	o
	2 (3G 7.2 Mbps)	o	o	o
	3 (WiFi 50 Mbps)	+	++	o
	4 (WiFi 50 Mbps)	+	+	o
	5 (LTE 50 Mbps)	o	o	+
Location 4	1 (LTE 7.2 Mbps)	+	++	o
	2 (3G 7.2 Mbps)	+	+	o
	3 (WiFi 50 Mbps)	o	o	+
	4 (WiFi 50 Mbps)	+	o	o
	5 (LTE 50 Mbps)	o	o	o
Location 5	1 (LTE 7.2 Mbps)	o	o	+
	2 (3G 7.2 Mbps)	+	o	o
	3 (WiFi 50 Mbps)	o	o	o
	4 (WiFi 50 Mbps)	+	++	o
	5 (LTE 50 Mbps)	+	+	o

Table 8: Device context based on location

After recording, the transferred video sequences were used to generate compositions of different streams according to the collected scores. Here, each generated sequence had a duration of 30 seconds, where switching between streams was allowed at most every 5 seconds. For each 5-second video segment, the device with the highest achieved score was selected as the video source. Figure 26 shows a still image from each recorded stream in location 4, where the first image is based on the recording of device three, having an average score of 3.64.

For comparison, we composed another video sequence of the same total duration randomly selecting one of the available views in 5-second intervals. Given the slightly different start time of the recording sessions, we normalized the time based on the arrival time of a stream on the server. The ten generated sequences (five score-based, five random) were then evaluated in a crowd-sourcing study using the Crowdee[aw] service, as shown in Figure 27. Both classes of generated sequences were displayed to users on mobile devices, gathering a subjective opinion score (i. e., MOS) using a continuous slider between 1 and 5, thus evaluating the overall

[aw] https://www.crowdee.de/

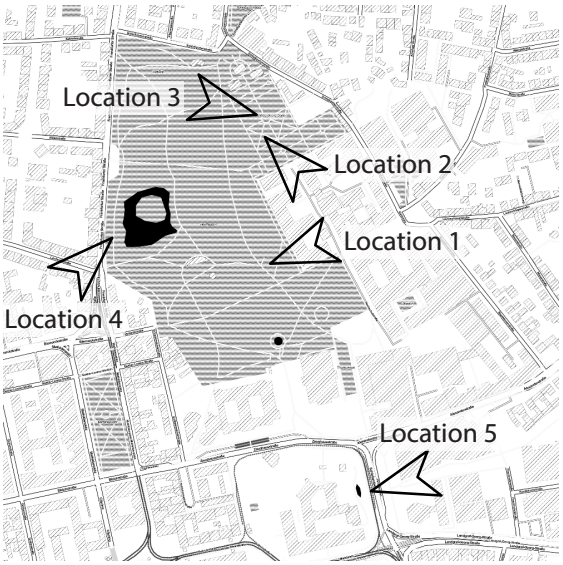


Figure 25: Recording location map for the field test used in the dataset generation,.

Figure 26: Example screenshots for location

quality of each sequence pair independently. The workers did not know which of the two sequences correspond to the context based recording-score composition.

After each pair of video sequences was shown to the workers, a forced-choice experiment was applied, asking which sequence they preferred. In a last question, the workers were asked to identify objects in the videos to verify a satisfactory level of attention during the test.

A second evaluation was conducted on a smaller scale, using two Samsung Galaxy S7 Edge devices, with the focus to observe the switching behavior delay during live video composition (see Subsection 4.4.3). Here, each device recorded a central visual display clock. Switching was induced between the two recorded streams while additionally recording the overall system behavior with a stationary camera.

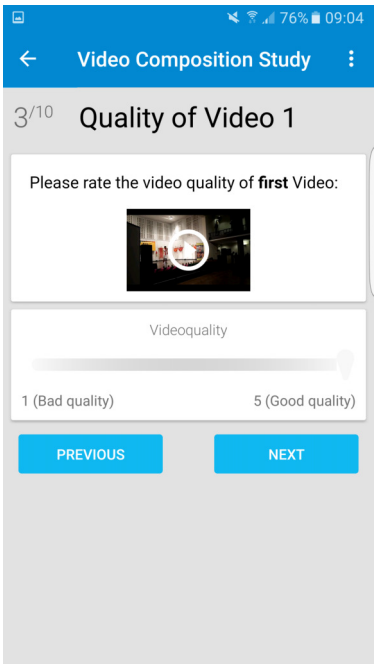


Figure 27: Screenshot of the user study in the Crowdee App presented to users. Playback was enforced on full screen and each video sequence had to be watched in full length. Content was prefetched to ensure uninterrupted playback.

4.4.2 *Crowd-sourced User Study on Quality of Experience*

The user study was conducted with 20 individuals, with an average age of 27 years (12 male, 8 female). Each subject passed a content recognition-based validation test, and all results were found to be usable by manual screening.

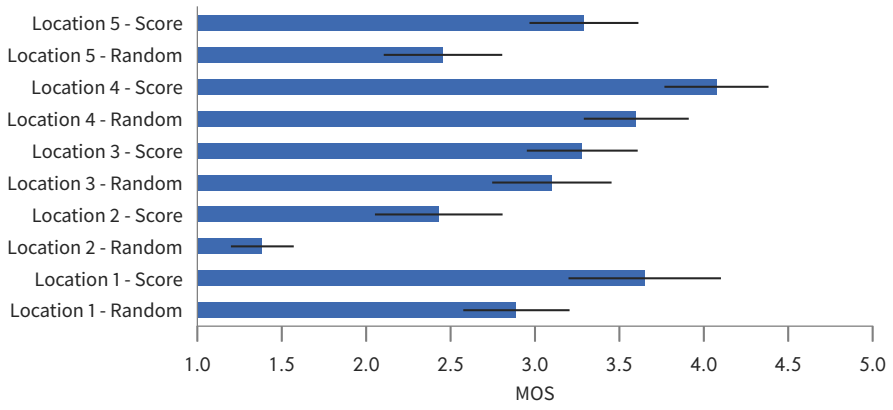


Figure 28: MOS by recording location as shown in Figure 25.

In Figure 28, MOSs are shown for each of the five recording locations. The entire spectrum of MOSs can be observed, indicating a heterogeneous quality and different user preferences in both random and score-based compositions. For each

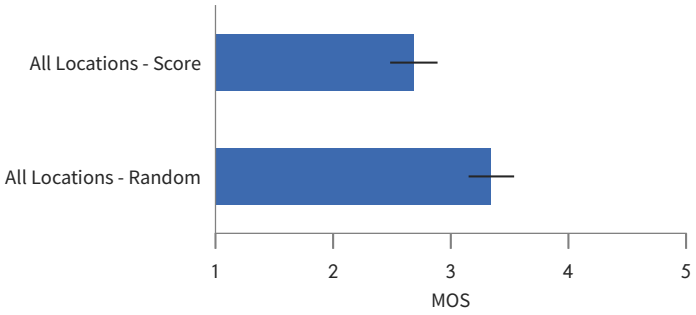


Figure 29: Combined MOS for all locations and users.

location, the MOS value for score-based compositions outperformed the random compositions. However, the overall difference in mean values strongly depends on the recording location. For locations 3 and 4 the confidence intervals indicate a wide spectrum of results, and no statistically significant preference for score-based over randomly selected compositions can be seen. Yet, there is a significantly better score for locations 1, 2 and 5.

A similar trend can be observed for the summarized results for all recording locations, as shown in Figure 29. Here, the score-based compositions show a higher mean, with both indicators showing a high standard deviation of roughly 0.5 on the MOS scale, indicating a high subjective variance in the preferences of users, with a significant preference of the score-based compositions.

The results for the JND-based forced-choice experiments in Table 9, as proposed by Watson et al. [WK01], show that three out of five score-based compositions are significantly better. This result verifies the trend seen earlier for the MOS-based

Video ID	% Score	% Random	JND unit
Location 1	88.9	11.1	1.7
Location 2	90	10	1.77
Location 3	66.7	33.3	0.65
Location 4	66.7	33.3	0.65
Location 5	78.9	21.1	1.18

Table 9: Just Noticeable Difference experiment results by recording location as shown in Figure 25.

results. Next, in Figure 30 we show the distribution of the results used for the score-based video composition. It can be observed that the overall weighted score has a mean of 3.5, where larger differences in the single indicators exist. The network score is never lower than 3, showing that results from the performed active measurements exceed the required bitrate for video upload. The distance score shows very low values overall, indicating in some cases imprecise location estimates or a high distance of the devices from the PoI. Last, the activity score shows a wider spectrum of values, which was expected given the range of disruptive actions performed by the recording users, as shown in Table 7.

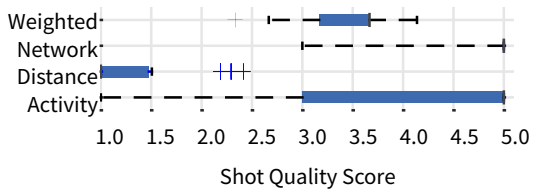


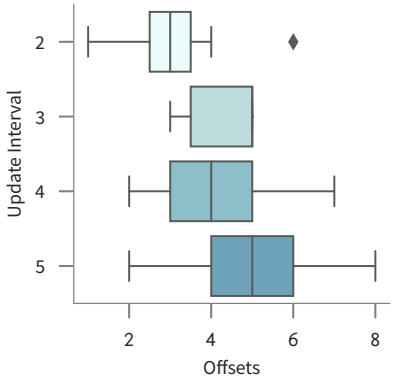
Figure 30: Comparison of generated scores for video selection.

4.4.3 Real-time Composition Capability

LiViU allows the real-time composition of the video streams using the monitoring framework to determine the devices that offer the best video quality. The server can use this information to play the video from the device that has the highest score.

In order to minimize the live playback drift and a potential stalling between switches, the server always retrieves the video streams from the two best devices. The packets from the first device are played continuously, and in case the stream stops, the server falls back to the device that has the second highest score. To analyze the efficiency of this approach, this evaluation scenario inspects the reaction time for users under lab conditions.

Given the focus on identifying the switching delay as well as the live playback drift, the factors for all parameters except the activity score were set to zero. For the update intervals, in which the devices transmit the current activity status to the server, the range between 2 and 5 seconds was tested in one-second steps. The reaction time for switching between the streams when shaking one of the two devices was analyzed (see Figure 31(a)). To accurately measure the user actions as well as the switching point, a central clock was recorded and later analyzed based on an external video recording, as shown in Figure 31(b). The results indicate that



(a) Distribution of the delay between triggering a switch by shaking and the actual update time for the composition

(b) Recording test setup

Figure 31: Process for generating end-to-end switch delay and distribution of results.

very low switching delays can be achieved with high update rates, although shorter update intervals also maintain an acceptable level of the overall switching delay. Regarding the required data for updates, each message has a size of approximately 380 Bytes. In addition, the bandwidth measurement uses 131 KBytes, initiated independently every 15 seconds. Thus, even for very short update intervals of 2 seconds, each device does not transfer more than 22.62 KBytes per minute for each `UpdateMesssage` plus an additional 524 KBytes for bandwidth measurement, including the complete set of indicators (all parameters have been transferred to the server but only the activity score was used for the switching).

4.5 Conclusions

This chapter presented an analysis of user behavior and video quality in the live-MBS YouNow and proposed a novel concept for the automatic generation of high QoE compositions of live User-Generated Video.

Based on the initial analysis, we derived that streaming quality is generally low and individual session times are often short in live MBS due to challenging network and device characteristics of uploading from smartphones. Thus, low-overhead selection of relevant content (i. e., high quality) is a crucial consideration for the live composition of User-Generated Video.

Motivated by these observations, we proposed a system for live upload and composition of UGV based on measuring the context data of the recording devices. We have presented a prototype implementation of our approach that was evaluated in a crowd-sourced user study. In particular, activity recognition and active bandwidth estimation were included as novel concepts to support mobile live video upload from multiple sources. Using the created dataset, we have shown that a higher mean QoE, as represented by the MOS and JND results, can be achieved when selecting video streams based on network-, activity-, and location-based indicators, *without analyzing the video content directly*. This allows the activation of only those devices that are used for the final view in a composed video stream, minimizing the overall data that needs to be uploaded. The fact that we do not perform video content analysis enables us to carry out the measurement and selection of devices as sources for the combined live stream in real-time with a very low end-to-end delay. We have shown that the generation of a combined stream is feasible in real-time with very low data transfer overhead while still allowing for quick switches between device streams.

Last, combining a context-based device selection with content analysis may provide high improvement potentials in the future. Here, in a first step, a subset of candidate devices can be identified for video upload using context indicators, while traditional content-based quality analysis can be used for detailed inspection of those streams. Thus, only a subset of devices that are likely to provide a high-quality stream needs to upload, while content analysis guarantees that features only visible in the content analysis are not disregarded. This would allow us to use most advantages of both approaches while resulting in a higher overall QoE.

5 Network Interdependencies in HTTP Adaptive Streaming

While the previous chapter discussed challenges regarding the upload of live-User-Generated Video, we now move our attention to the other end of the distribution chain of today's Video Streaming Systems (VSSs). Here, in the delivery of high-quality video, the Over-The-Top (OTT) concept has become the central mode of distributing Video on Demand content to users; while this provides great flexibility, achieving consistent Quality of Experience (QoE) is challenging given the heterogeneity of underlying networks used in Internet-based delivery.

In particular, the interdependence between network characteristics and configurations poses a significant challenge when trying to identify the causes of low QoE. For example, if the given selection of bitrates by the Adaptation Algorithm (AA) in a sessions leads to an undesirable experience for the user (e. g., the video stalls, the quality is low or often drops significantly) this leads to the question of what is the underlying reason for this insufficient performance. Here, the research focus has been on investigating the AA and the underlying decision model itself. However, given the layered architecture of Dynamic Adaptive Streaming over HTTP (DASH), we argue that there is sufficient reason to further investigate the network and configuration space beyond AAs, e. g., *how can the AAs performance be explained in the full system context*.

The underlying, crucial observation here is that to consider the performance of DASH sessions, all interdependent system components—including players, Congestion Control (CC), and execution environment—influence the performance.

With the work presented in the following chapter, we address these challenges by investigating DASH under realistic assumptions towards achieving a tangible (and measurable) increase in user satisfaction—or QoE —in today's prevalent DASH-based VSS by conducting and analyzing large-scale, reproducible experiments in a controlled test environment.

To this end, we first address the underlying motivation and challenges involved with a large-scale analysis of DASH. We then proceed to detail the approach that allows investigating a large-scale evaluation space to identify such configuration trade-offs that we call *sweet spots*. In the following evaluation, we investigate such concrete relations between Quality of Service (QoS) and Quality of Experience (QoE) indicators with the network environment and configuration space for DASH. Based on this broad range of DASH configurations and trade-offs, we then focus on learning approaches to identify the best context-dependent set of such configurations by selection of players, AAs, and CCs for a given environment context.

The presented analysis is based on our previously published work in [Sto+16b; Sto+17a].

5.1 *Extensive Emulations for DASH*

Our central research claim is that the viewers' QoE depends on the configuration of the application and the network conditions in a complex, non-trivial fashion. Thus, we propose to evaluate DASH applications, the impact of their configuration and the network conditions on QoE metrics in large-scale emulations. Given the stated motivation, this requires to cover an extensive evaluation space to explore DASH configurations and the effect of *combinations of configurations*, as well as a systematic approach to makeing the scale of such experiments feasible. In the process of following this goal, a general network experimentation framework called *maci* emerged that is covered in detail in Frömmgen, Stohr, Koldehofe, et al. [Frö+18]. Aspects of this systematic approach are the result of the first part of our DASH studies [Sto+16b]; in Stohr, Frömmgen, Rizk, et al. [Sto+17a] *maci* was the framework used to conduct experiments.

We begin this section with an outline of the architecture of *maci* that supports this large-scale emulation and experiment design. We then describe the design of encapsulated DASH emulations using `Mininet`, followed by a discussion of the emulation context, e. g., with regard to the environment and the configuration space.

5.1.1 *Evaluation Framework Architecture*

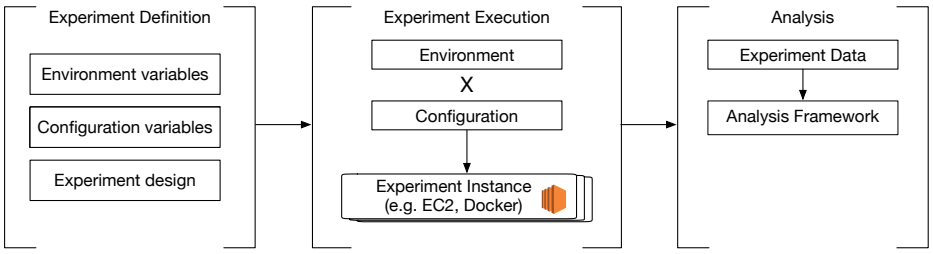


Figure 32: Extensive emulations with real applications on multiple network conditions.

The presented approach was developed in conjunction with *maci* by Frömmgen, Stohr, Koldehofe, et al. [Frö+18]. It is an integrated extensive emulation environment for conducting networking experiments, along with specific components for DASH studies. By using this approach, we intend to overcome several challenges usually faced in conducting networking experiments: *1)* increasing complexity; *2)* increasing innovation speed; *3)* the need for extensive experiments; *4)* and limited resource availability . Figure 33 shows the architecture used to emulate the Cartesian product of combinations of environment and configuration variables (see Table 10) for the DASH studies in this work (1). On a high level, the system consists of three components. First, a web-based interface allows defining experiments in an abstract form by selecting the desired set of environment and configuration variables that can be instantiated within a concrete experiment

design for the given use case (2). These instances are then scheduled, given the available resources, in parallel (3), as concrete experiment instances. Here, any virtualization concept, such as virtual machines or container-based virtualization, may be used to execute the defined experiment (4). Results generated by these experiments are then collected and aggregated for further analysis after which the can be used as a basis for refinement of the initial experiment parameters interactively. This last component uses interactive `Jupyter Notebooks` and the `SciPy` environment for flexible data analysis. For a detailed description of this concept, we refer to [Frö+17; Frö+18]. In the context of this work, we developed a custom execution environment for DASH experiments, as described in the following.

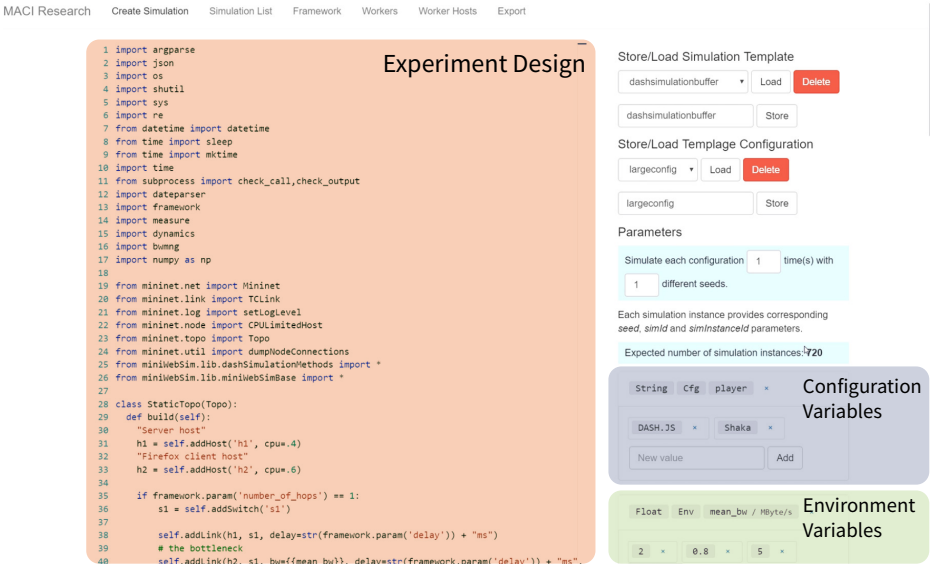


Figure 33: Example screenshot of the experiment definition interface.

5.1.2 DASH Emulation Environment

Driven by the goal to compare DASH players in a reproducible setup, we build a player execution environment that provides a *player abstraction* for automatically loading, configuring and monitoring, JavaScript, as well as Python-based players, as illustrated in Figure 34. The first setup layer of the experiment is built using Python and relies on `Mininet` [Han+12] to configure a virtual network setup consisting of a server, a client and one or multiple bottleneck links separated by switches and shaped with respect to the current network environment configuration. This enables future expansion of the experimental setup to arbitrary topologies and more complex networking conditions.

On the server-side, a web server is launched using the Node.JS *http-server*[ax] package or Apache, while the client starts a `Xvfb`[ay] session to allow for the execution

[ax] https://www.npmjs.com/package/http-server
[ay] https://www.x.org/archive/X11R7.6/doc/man/man1/Xvfb.1.xhtml

of the *Firefox 52* browser that supports Media Source Extensions.[az] Further, the *Geckodriver*[ba] service provides control of the browser by Selenium-API calls from the Python experiment description code.[19,bb] For the JavaScript-based DASH players (*DASH.JS* and *Shaka Player*), the experiment is loaded into the browser by requesting the web page containing the DASH player's JavaScript application after the setup of the DASH player execution environment. Here, the Webpack[bc] loading system is employed to dynamically instantiate the selected DASH player based on the configuration parameters. For each player, a loading script was implemented that maps initialization routines and settings such as buffer levels, the Media Presentation Description (MPD), and Adaptation Algorithms to corresponding function calls. Since AStream, in contrast to DASH.JS and Shaka Player, is built entirely in Python and *does not* play out the video, it is directly called with the corresponding experiment parameters without browser interaction.

[19] *Now there is also the option to use Chrome-headless as an alternative to the `Xvfb`-based approach.*

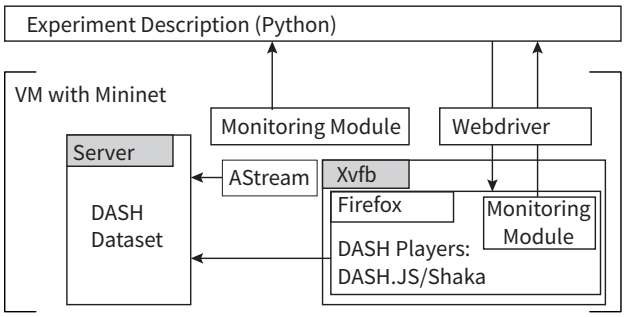


Figure 34: Experiment instance as represented by one virtual machine.

Monitoring Performance Metrics

To retrieve and analyze the system events and data structures which vary depending on the selected player, we implemented a *monitoring module*. It maps the players' monitoring events to a set of callback functions and creates a universal data structure saved as a JavaScript array. We provide modules for the JavaScript and Python based players respectively.

We set the play out duration for each experiment to 120 seconds and continuously monitor the playtime. This is to ensure that, even if stalling events occur, players are compared on a playback time instead of an experiment time basis. After the given experiment time, we collect and process the experiment metrics.

In this work, we consider two metric types to assess the performance of DASH Players: *1)* target metrics that are directly *measured*, *2)* and aggregate metrics that are derived from target metrics.

[az] http://caniuse.com/#search=mse
[ba] https://github.com/mozilla/geckodriver/
[bb] https://developers.google.com/web/updates/2017/04/headless-chrome
[bc] https://webpack.js.org/

5.2 Cross Layer Environment and Configurations

Analog to the above-given motivation of a large-scale emulation approach, we selected a wide range of parameters that are presented in a bottom-up fashion, as specified in Table 10. In the following, we provide details for each category.

5.2.1 Network Conditions

As the choice of a wide range of realistic network conditions is fundamental for allowing insights into the dependencies between configurations, we consider two synergistic approaches: *i) synthetic conditions* as a model to derive a random (but reproducible by a given seed) bandwidth and delay distribution specified by mean and variance, and, *ii) dynamic real-world bandwidth traces* collected from mobile devices [Rii+13], and replay these using a `tc` traffic shaper. We argue that meaningful insights require both approaches.

The additional range that can be achieved by synthetic distributions allows a systematic comparison in extreme (e. g., low mean bandwidth, high variance) and more static conditions (e. g., low bandwidth variance), whereas recorded, dynamic bandwidth traces allow to analyze the impact of fluctuating bandwidth conditions as they are experienced in reality. For all bandwidth trace types, we further specify additional variations in the delay and loss.

5.2.2 Transport Protocol

As DASH uses HTTP and TCP, the TCP congestion control algorithm influences the achievable throughput and interacts with DASH's adaptation loop. We investigate three TCP congestion control algorithms: First, we include TCP Cubic that adjusts the congestion window size (cwnd) based on a cubic function for high bandwidth utilization. In contrast, TCP New Reno is based on the additive increase, multiplicative decrease concept for cwnd control, adding the concept of fast recovery to its preceding implementation TCP Tahoe. Last, we include TCP Vegas because it is sensitive to changes of the round trip time (RTT) and adjusts the cwnd by detecting changes in the RTT. From the selection of these concrete CCs we expect varying influence on the performance given their fundamentally different design principles. A more detailed discussion of Transmission Control Protocol (TCP) CC is given in Section 2.1.

5.2.3 Adaptation Algorithms

Given that DASH is a standard for client-driven video streaming with multiple media quality representations, the client player requires a logic (often denoted as Adaptation Algorithm (AA)) to decide on the requested media bitrate and quality. The DASH AA can be classified into three main categories: *i)* Throughput-based Adaptation (TBA), *ii)* Buffer-based Adaptation (BBA), *iii)* and Hybrid Approaches. We discuss these AAs concepts in detail in Subsection 3.2.1.

In this presented study, approaches in the first two categories are implemented by the used DASH players. Here, DASH.JS and AStream implement the BBA Buffer Occupancy based Lyapunov Algorithm [SUS16]. Further, TBA is provided

	Zone/Layer	Configuration	Instantiations	#Mechanisms
Configuration Variables	Video Representation	Segment Size [s]	1, 2, 4, 6, 10, 15	5
	DASH Player	Buffer Size	Default, 5, 10, 20	4
		Execution Enviroment	DASH.JS, Google Shaka-Player, AStream	3
		Adaptation Algorithm	TBA_3, TBA_{10}, TBA, BOLA,	2
	Transport Protocol	Congestion Control	Vegas, Cubic, Reno	3
Environment Variables		Bandwidth	Dynamic: Bus, Car [Rii+13]	3
			Generated (based on seed, mean and variance) $\mu_{BW} 0.8, 2, 5, 7.5, 10$ [Mbps], $\sigma^2_{BW} 0, 0.8, 2, 5$ [Mbps]	20
		Latency [ms]	10, 20, 35, 50, 100, 150	6
		Packet loss [%]	0, 0.5, 1.0, 2.0, 5.0	5
		Hops	1, 2	2

Table 10: In two separate studies we explore an extensive set of combinations in the configuration and environment space of DASH.

by DASH.JS and Shaka Player. In case of Shaka Player, we investigate varying configurations of the algorithm by modifying the considered measurement interval. By including both concepts, as represented in different players, we investigate specific characteristics of TBA and BBA as these algorithms may comprise different *aggressiveness* in their quality adaptation. For example, cautious adaptation algorithms that try to avoid stalling events choose conservatively low qualities in the downscaling direction, i. e., when choosing a lower quality than the previously requested video segment. The same conservatism holds in the upscaling direction, i. e., when increasing the quality for one of the requested segments. Derivations of the details and mechanisms within TBA and BBA algorithms usually resort to modeling assumptions in players that we discuss in the following.

5.2.4 DASH Players

In the following, we briefly discuss the open-source DASH players that we analyze and compare in this work. While many other DASH players exist, such as the ones used by Netflix and YouTube, the lack of an open API prevents their direct use in the context of this work. Other commercial players that provide an API, such

as Bitmovin's *HTML5 Player*[bd] could be considered in our work, but have been omitted due to licensing reasons.

DASH.JS is the reference implementation by the DASH Industry Forum. The release v2.3.0, used in our experiments, features two types of AAs: TBA (default) and BBA. The TBA uses a `ThroughputRule` that maintains an up-to-date list of the previous four throughput and latency measurement periods that determine upcoming adaptation decisions by passing them on to a `SwitchRequest` class. The BBA algorithm embedded in DASH.JS features *BOLA* as introduced in Subsection 3.2.1.
Configuration: DASH.JS comprises configuration parameters for both BOLA and TBA that include the target buffer levels depending on the current playback state. If the player selected the top quality, a target buffer size of 30 seconds (`BUFFER_TIME_AT_TOP_QUALITY`) is used. Otherwise, a smaller target buffer size of 12 seconds (denoted `DEFAULT_MIN_BUFFER_TIME`) applies.

Shaka Player that is developed by Google[be] features a TBA algorithm that conservatively uses the minimum of two exponentially weighted moving average (EWMA) variables derived from throughput measurements within a period of two and five seconds, respectively. This aims at a faster downscaling in case of bandwidth drops and a slower upscaling with increasing bandwidth measurements.
Configuration: To evaluate the design decision for Shaka Player's AA, we added two EWMA-based Adaptation Algorithm (AA) to the default configuration, which only relies on three (TBA_3) or ten (TBA_{10}) segments for calculation. Further, two other core settings control the playback buffers in the Shaka Player. First, the `rebufferingGoal`, which is set to 2 seconds, provides the minimal buffered video duration before the playback is started. Second, the `bufferingGoal` of 10 seconds provides the maximum buffer level that is loaded in advance. With the goal to compare fast as well as very conservative adaptation decisions, in addition to the default AA settings, we modified parameters in the `SimpleAbrManager`-class `MIN_SWITCH_INTERVAL=5`, default 30; `MIN_EVAL_INTERVAL=1`, default 3) to allow for faster adaptation, given the 120-second test sequences.

AStream is a *headless* Python-based player that is introduced in [JTM16] to simulate DASH players, i. e., it is not able to natively play out the requested video segments. AStream served as a rapid prototyping environment for multiple TBA and BBA Adaptation Algorithms such as [JTM16; WRZ16]. In this work, we consider an implementation of BOLA within AStream in comparison to the BOLA implementation within DASH.JS. The rationale behind this comparison is to illustrate the impact of the player environment while keeping the AA fixed.
Configuration: As with the other introduced players, *AStream-BOLA* has a predefined default target `bufferlevel`, here it is set to 15 seconds. AStream

[bd] https://bitmovin.com/html5-player/
[be] https://github.com/google/shaka-player/releases/tag/v2.0.6

also comprises additional configuration parameters for which we refer to the source code [JTM16].

5.2.5 *Video Content*

The video content in our tests is based on the open movie *Tears of Steel*[bf] provided in a dataset prepared by [LMT12]. It consists of nine H.264-AVC encoded video representation layers having bitrates between 0.253 and 10 Mbps with resolutions between 480×270 and 1920×1080 pixels in 2s and 10s segment size configurations, offering similar resolutions and bitrates as used in practice by YouTube, Netflix and Apple [Kre+16].

Table 11: DASH dataset representations

Resulution	Bitrate (Mbps)	FPS
1920×1080	10.0, 6.0, 4.0, 3.0	24
1280×720	2.4, 1.5	24
480×270	0.807,0.505, 0.253	24

5.2.6 *QoS and QoE Metrics*

First, we consider the set of the following directly measurable QoS metrics: *i)* the initial playback delay, *ii)* the total stalling duration, *iii)* the initial playback bitrate, *iv)* the average playback bitrate, *v)* the average number of adaptations, *vi)* the download duration and length of requested segments, and *vii)* the average amplitude of adaptations. These QoS metrics also build the basis for aggregate performance measures (i. e., QoE indicators). Here, we employ a model introduced by Hoßfeld, Schatz, et al. [Hoß+13] to derive a Mean Opinion Score (MOS) score based on stalling frequency and length, denoted as $\text{MOS}_{\text{Stal.}}$. Next, as a means to evaluate the video quality, we apply the model presented by Hoßfeld, Seufert, et al. [Hoß+14b], providing a MOS based on the portion during which the session stayed in the highest quality representation. We will refer to this metric as $\text{MOS}_{\text{Rep.}}$. As the proposed models are based on 30-second test sequences, we derive the average of four independent scores for each 30-second fragment of our 120s test intervals. For reproducibility, we use the same dataset as originally used to derive the $\text{MOS}_{\text{Rep.}}$ model in [Hoß+13] (c.f. Subsection 5.2.5). Unlike the original work presenting this model, we also include quality representations exceeding 0.807 Mbps (having a resolution of 640×360 pixels), classified as high-quality layers, to explore the potential for high-bandwidth configurations. For a detailed discussion of these metrics, see Section 2.4.

5.3 *Evaluation Concept for DASH*

This section presents evaluations using our proposed QoE-driven extensive emulation approach for DASH. Here, we rely on two datasets. The first dataset

[bf] https://mango.blender.org/

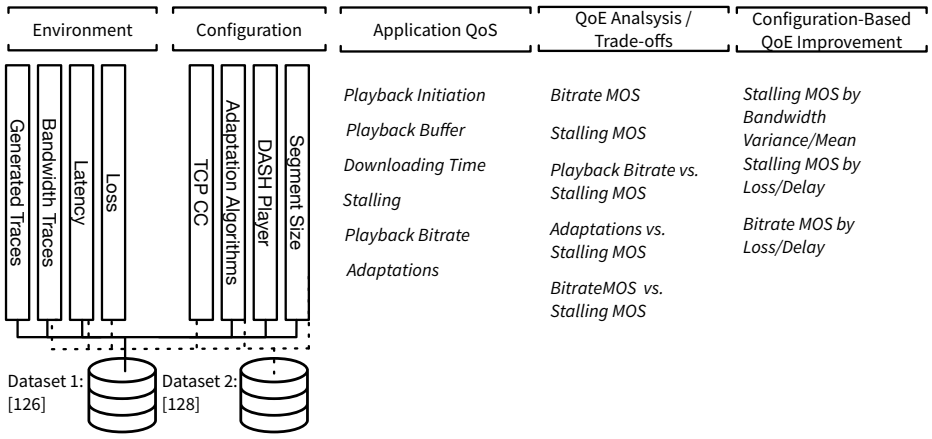


Figure 35: Overview of the evaluations conducted by metric along the axis of environment conditions considered.

addresses ([Sto+17a]), in particular, the influence of DASH players, by example of DASH.JS and Shaka Player, and considers a vast array of recorded and synthetic dynamic traces to emphasize the particular influence of players, buffers and AAs while reducing the focus on very low-quality network behavior with delays and losses. The second dataset ([Sto+16b]) emphasizes the effect of TCP CC in dynamic traces and includes particularly detrimental delays and losses, as potentially experienced in mobile scenarios while reducing dimensionality by considering a single player. Nonetheless, each dataset is independently covering an extensive array of combinations spanning all relevant layers, as shown in Table 10.

In Figure 35, we show the categorization of this analysis in different areas; we begin with application-centric QoS factors that show the underlying differences in performance and behavior of context and configuration parameters as exposed by our dataset. We then continue to explore such performance aspects based on existing QoE models that map the before-mentioned service attributes to concrete measures regarding their significance on the service quality for users. In particular, we employ two MOS models, one related to playback interruptions and the other one concerning the bitrate quality.

Using this underlying understanding of the effects of such mechanisms, including their influence on users, i. e., concrete cross-layer configurations, we extend our findings by presenting them in a comparative QoE analysis, considering a diverse set of network conditions to point out performance affinities.

This analysis is extended to present MOS trade-offs by identifying Pareto optimal configurations of concrete DASH multi-mechanisms. Last, we derive models for QoE-based configurations based on environment conditions.

5.3.1 The Case for an Emulation-based Evaluation of DASH

As a prerequisite for the following case, we introduce key underlying assumptions in DASH Adaptation Algorithms. Generally, these algorithms take network connection measurements and internal player information to *decide* on video segment requests to optimize playback quality. In its most general form, this adaptation (or

decision) problem can be modeled as a stochastic control problem that is, however, cumbersome to formulate let alone to solve online.

Adaptation Algorithms that fall into the TBA or BBA classes usually adopt assumptions on the behavior of the network and the media player that allow analytical optimizations, or at least the development of online heuristics. A common assumption is a stationary random process for the network behavior, e. g., regarding the available bandwidth process that governs the streaming connection. For example, given this assumption, BOLA provides approximately optimal decisions on the segment requests with respect to a QoE utility metric. A further assumption that is often adopted is that network observations are (conditionally) independent of the network state, i. e., the measurement samples at time i for segment n $\{n_t \mid t = i\}$, are not dependent on previous network states $\{n_t \mid t < i\}$. This confines the need for modeling network correlations, however, this may be a strong assumption, especially in the presence of bursty traffic. Given this assumption, heuristics can be derived, e.g., employing Kalman/particle filter types of control for TBA algorithms. A basic example of such control is the EWMA employed in the Shaka Player.

It is, however, important to note that models for AA do not capture the impact of player-specific components, e. g., additional processing delays and the method of measurement by the player itself. AAs assume a certain statistical quality of these network measurements. However, the quality of the network measurements can be strongly affected by the way the player requests new video segments [WRZ16]. A striking example is given by the simulative DASH player AStream originally

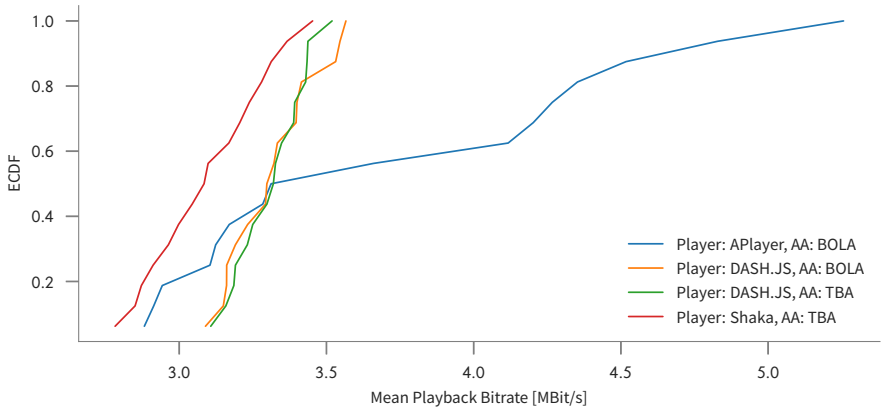


Figure 36: Cumulative distribution over mean bitrate of entire sessions in a comparison between JavaScript-based players (DASH.JS and Shaka Player) and AStream for a given environment bandwidth of 5 Mbps (variance ± 0.8 MBytes). AStream shows a distinctly different distribution given the wrong interpretation of segment sizes in adaptation decisions.

included in our work. Figure 36 shows that, for a mostly static environment, there is a vast difference in the mean playback bitrate between the JavaScript-based players DASH.JS and Shaka Player, and AStream. While the former two players show a similar distribution in mean sessions bitrates between 3 Mbps and 3.5 Mbps, given the environment bandwidth of 5 Mbps (variance ± 0.8 Mbps),

AStream has an unrealistically high mean session bitrate for most sessions, with roughly 10 % being even higher than the environments bitrate. Given the measured inconsistencies of the results observed for AStream also present in initial delay and stallings (cf. [Sto+17a], Table 16), we reason that the absence of an actual playback buffer[20] in *simulative playback* leads to potentially unrealistic assumptions in AStream's performance. For example, an *immediate* start of the video playback once *any* stream component is fetched in AStream can be explained by the way the `init` segment is processed. Depending on the dataset, the `mp4` `init` segment contains the `moov`[bg] element that describes stream meta-data, but does not contain video frames. It should not lead to playable content becoming available to the player. However, in our tests, fetching these segments did lead to an increase in AStream's buffer level.[bh] Given that these `init` segments are respectively small in file size compared to segments containing frames, as represented by `m4s` segments, the buffer is falsely assumed to be filled very fast when initiating the stream. This obscures segment downloads and impacts initial adaptation decisions as the (false) high buffer filling leads to undesired up-adaptations, as well as stalling at the playback begin. In contrast, both JS players exhibit the expected behaviour and begin playout after the first `m4s` segment is downloaded. Given these inconsistencies, we have excluded AStream in other analyses of this evaluation.

[20] *such as the MedieSourceExtentions API providing the video playback buffer in web browsers*

5.3.2 DASH QoS: Cross-Layer Performance Characteristics

The first part of our evaluation covers analysis based on QoS metrics, as an indicator of concrete performance differences between configurations. For an overall reference of the global results across the full respective dataset for both simulation classes, we present the used indicators in Table 16. Here, the mean and standard deviation are noted for each player and AA combination across all QoS metrics.

Table 12: Overall comparison of adaptation, video quality, and stalling metrics.

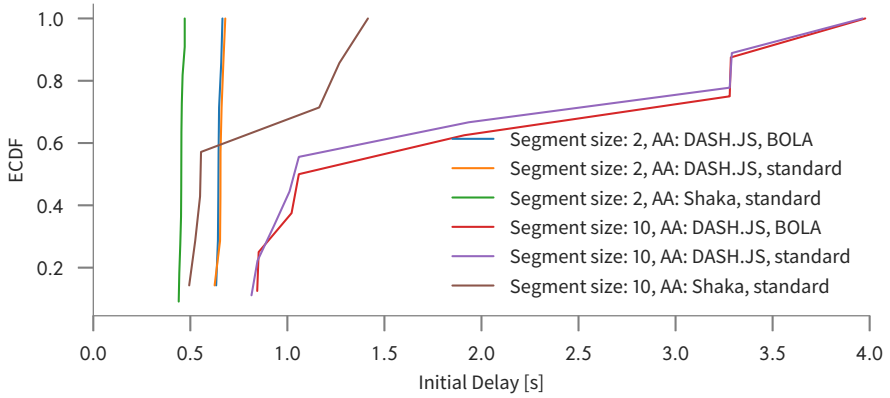
	Dataset 1								Dataset 2					
	DASH.JS TBA		DASH.JS BOLA		Shaka Player TBA		AStream BOLA		Shaka Player TBA		Shaka Player TBA_3		Shaka Player TBA_{10}	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Init. Rep. [Mbps]	0.8	0.0	0.8	0.0	0.3	0.0	0.3	0.0	0.8	0.0	0.8	0.0	0.8	0.0
Init Delay [sec]	1.6	2.4	1.6	2.3	0.7	0.7	0.1	0.0	3.3	5.5	3.5	6.2	3.3	5.7
Adaptations [#]	8.6	8.5	8.6	8.5	4.8	2.8	18.2	13.7	2.7	2.0	2.9	2.0	2.3	1.5
Amplitude [level]	1.3	1.0	1.3	1.0	1.7	1.9	1.7	2.4						
Stalling sum [sec]	8.8	14.9	8.9	15.0	8.3	13.8	18.0	50.3	26.3	41.6	28.6	46.7	28.7	46.0
Stalling avg. [sec]	1.2	1.8	1.2	1.8	1.1	1.8	4.1	11.3	7.5	16.3	12.0	28.1	12.3	27.9
Stalling [#]	4.1	3.8	4.1	3.8	3.7	3.5	3.5	7.9	3.4	5.5	1.7	1.8	1.7	1.8

[bg] http://l.web.umkc.edu/lizhu/teaching/2016sp.video-communication/ref/mp4.pdf

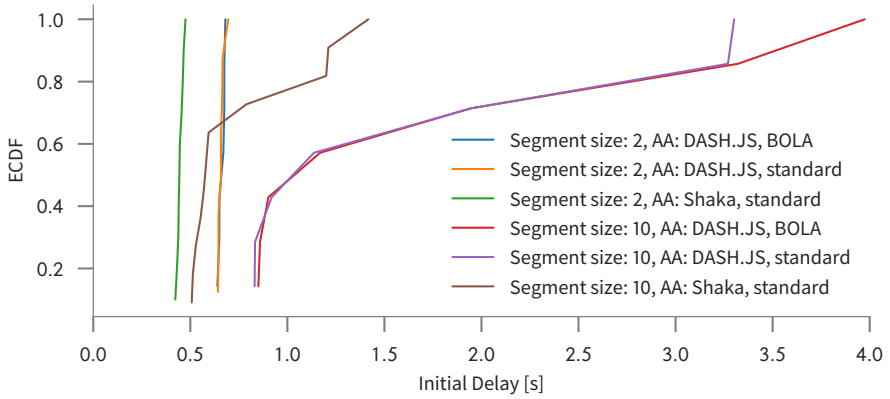
[bh] https://github.com/pari685/AStream/blob/master/dist/client/dash_buffer.py#L129

5.3.2.1 *Initial Delay*

The initial delay, i. e., the time until rendering the first frame, is an important performance indicator for DASH streaming sessions since longer initial delays are known to have a negative impact on the viewers' engagement [Dob+13]. Yet, minimizing these delays usually corresponds to a lower initial playback quality. Thus, it is crucial for players to strike a balance between these opposing factors.



(a) Initial delay for default buffer setting



(b) Initial delay for 20s buffer setting

Figure 37: The initial delay depends on the segment length and the used player. Neither the buffer size nor the Adaptation Algorithm severely influences the initial delay.

Figures 37 and 38 give insights into these design choices by showing the empirical distribution function of the initial playback delay vs. different combinations of players, adaptation algorithms, buffer sizes, segment lengths and the underlying TCP CC. Here it is evident that three main factors impact the initial delay: *i)* the segment length, *ii)* the used player, *iii)* and the CC. Beginning with the segment length, the distributions on the Empirical Cumulative Density Function (ECDF) show similar trends along the segment length setting, i. e., for 2-second segments

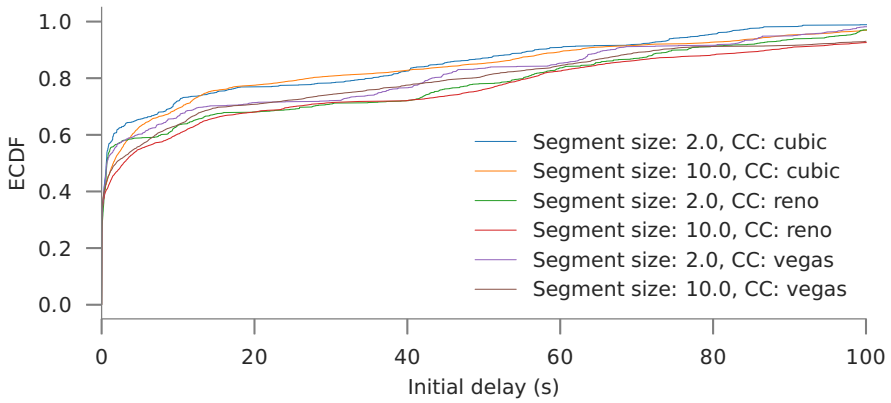


Figure 38: For very diverse network conditions, including high losses and delay, there is a long tail of high initial delays. However, TCP cubic reduces the overall initial delay.

the distributions are clustered around 0.5 seconds whereas 10-second segments are distributed widely between 0.5 and 4 seconds. The impact of the player is apparent when comparing DASH.JS' results with Shaka Player where initial delays vary independently of Adaptation Algorithms. Our analysis indicates that the buffer size does not impact the initial delay. This is expected since the target buffer size is usually used to gauge the steady-state adaptation behavior of the player while many players have only minimal buffer level requirements to start playback.

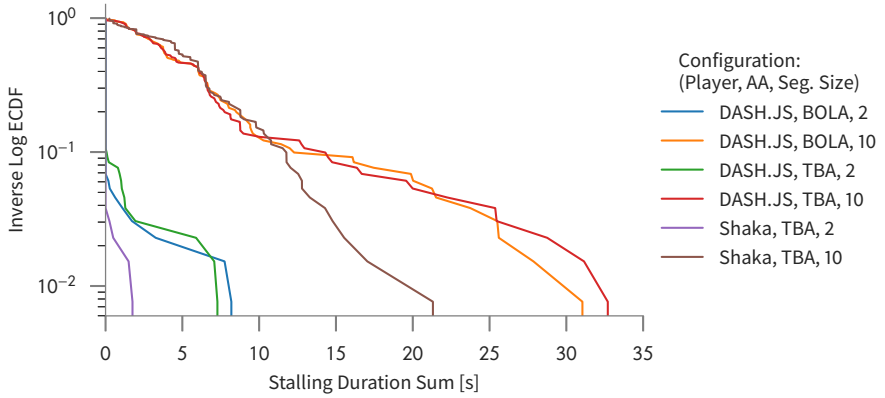
A fundamental characteristic that drives the differences in the observed initial playback delays is the selected initial representation. For example, DASH.JS consistently selects the third lowest representation, while Shaka Player begins playback with the lowest representation, leading to smaller download sizes for the initial segments. This is reflected in the first two rows of Table 16. However, accepting longer initial delays allows for a higher initial playback bitrate as indicated in Table 16. Thus, this represents a design choice of the player as to which layer's segment is to be downloaded in the beginning before any measurements are passed to the AA.

Last, as shown in Figure 38, for more varied network contexts with high losses and delays, initial delays are in case of Shaka Player extremely long given the low network quality in some sessions. Yet, there is a clear trend, that over all sessions, TCP Cubic has a lower initial delay, indicating that beyond the choice of the player, TCP CC influences initial delays.

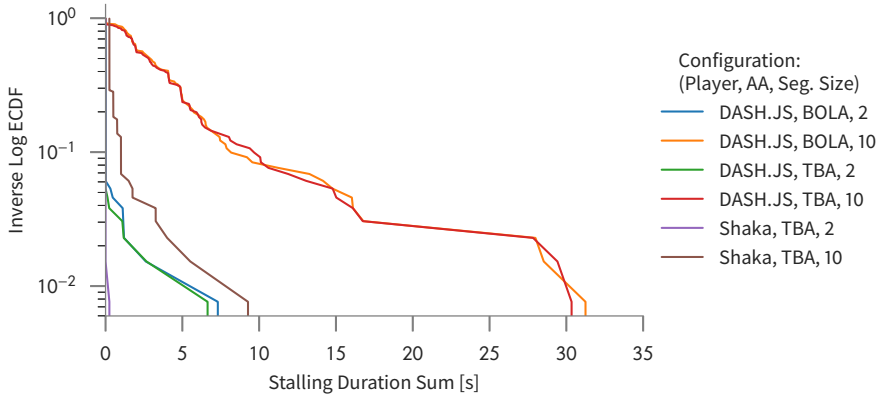
5.3.2.2 *Stalling*

Figure 39 depicts the CDF analysis of the total stalling duration during the playback of the 120-second video for different players, Adaptation Algorithms and target buffer sizes across all network bandwidths environments.

The players show a distinct and consistent pattern where the configured segment size predominantly influences the total stalling duration so that smaller segments significantly reduce the stalling duration.



(a) Using the default buffer configuration for the respective players across all environment conditions.



(b) Increasing the buffer size to 20s across all environment conditions.

Figure 39: Impact of the buffer size on the total stalling duration. Players differ significantly while Adaptation Algorithms show a small influence. Larger buffers, i. e., 20 seconds, reduces stalling in half for 80 % of the sessions with large segment sizes.

Further, we see the influence of the buffer size, in particular, when comparing the 50 percent percentile between Figure 39(a) and Figure 39(b); this corresponds to roughly 3 and 5 seconds of stalling time respectively. In general, Shaka Player outperforms the other alternatives with nearly no stalling events for large buffers and small segments sizes. Surprisingly, we note that the Adaptation Algorithm has a minor impact on the overall stalling duration when compared to the target buffer size.

Regarding the CC, we again see an overall better distribution, in terms of less stalling time, for all configurations using TCP Cubic, as shown in Figure 40.

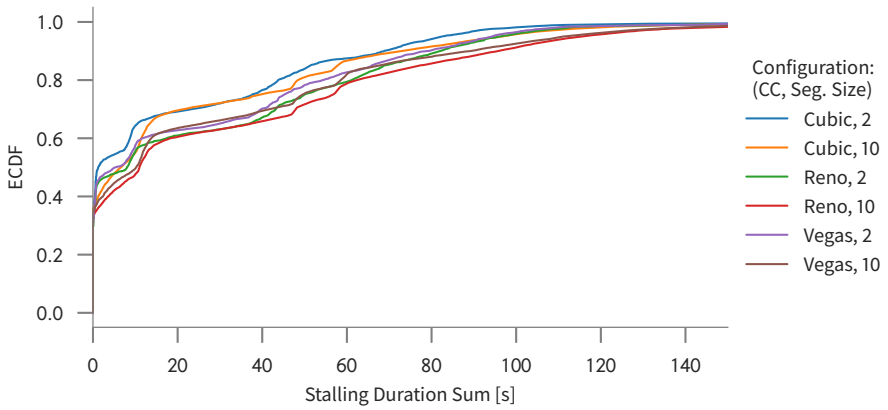


Figure 40: Impact of the CC on the total stalling duration.

5.3.2.3 *Downloading Behavior and the Relation with Segment Size and Buffers*

Figure 41 compares the distribution of buffer fill levels across all configurations between 2s and 10s segment sizes. In both cases, the buffer fill state is strongly cumulated close to zero, which is to be expected given that a large part of the tested network settings lead to significant stalling, e.g., when bandwidths are low or the packet loss rate is high. A second bend can be seen for the 2s segment configuration caused by the player having loaded exactly one segment from the initial or buffer under-run state. For full buffers, there is a clear distribution

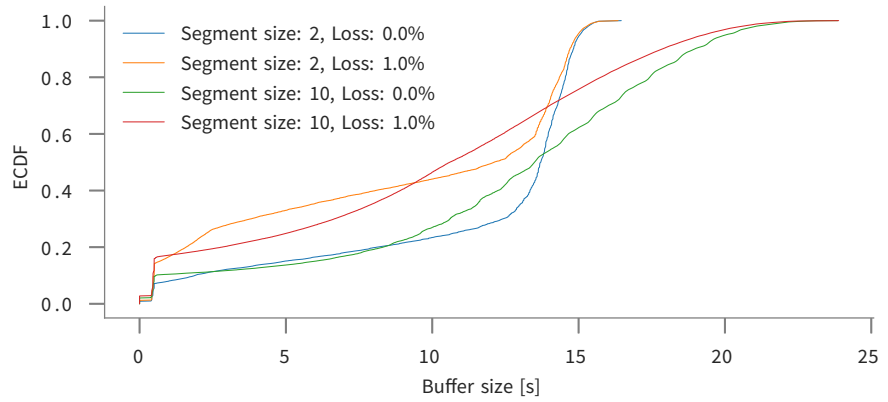


Figure 41: Distribution of the DASH player buffer level depending on the segment size

difference past the 60% density between 2s and 10s configurations, having the range of buffer fill states spread out in the 10s case whereas 2s segments have a high cumulation rate around the fixed 15s upper bound for the buffer level in the Shaka Player. For 10s segments this difference can be explained based on their higher segment length compared to 2s segments in relation to the total buffer size of 15s, which allows them to overfill the players buffer. However, the median buffer level is still higher for 2s segments which is somewhat counter-intuitive.

Regarding observations made from dataset 1, we note that if the target buffer size is small the player often withholds requests due to a full buffer. DASH players are known for an ON-OFF behavior while downloading video segments which arises due to adaptation algorithms. This behavior is reinforced by a consistently filled buffer due to a small buffer size relative to the segment duration length. Thus, the player is blocked from requesting new segments which is evident in Figure 42, where the player downloading state exceeds 100 seconds for a large target buffer size paired with 2-second segments. In contrast, the player has a significantly lower mean downloading time, i. e., 75 seconds for a low target buffer size paired with 15 second long segments. However, potentially unlimited buffers can have a negative

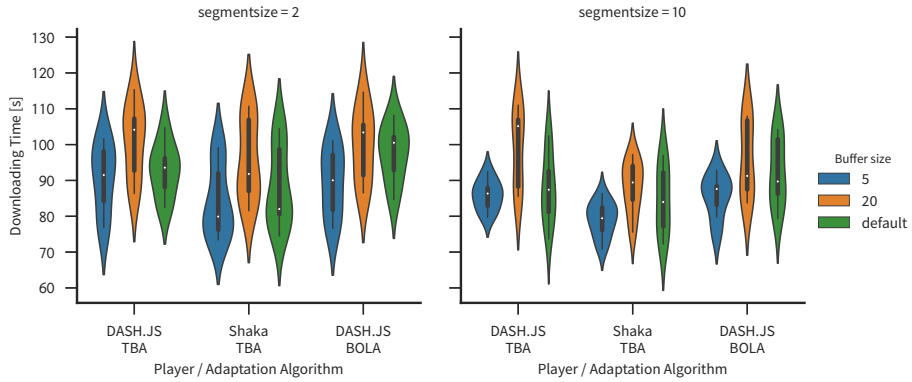


Figure 42: Impact of the buffer and segment length on the total time spent in downloading state. Respective shapes show the median and distribution by player, AAs, and buffer size. Inner black boxes represent median and IQR, with extending lines indicating the full distribution span. Further, the surrounding areas indicate this distribution as a KDE.

influence on the playback quality. In case of high buffer levels, the entire buffered content is played out, and more time may pass until the quality is adjusted, i. e., when the bandwidth increases significantly mid-session. Alternatively, segments that have already been fetched to the buffer may also be discarded[21] leading to a potentially high wastage of buffered content. Especially in a mobile context, where data caps are still prevalent, this translates to high costs for users.

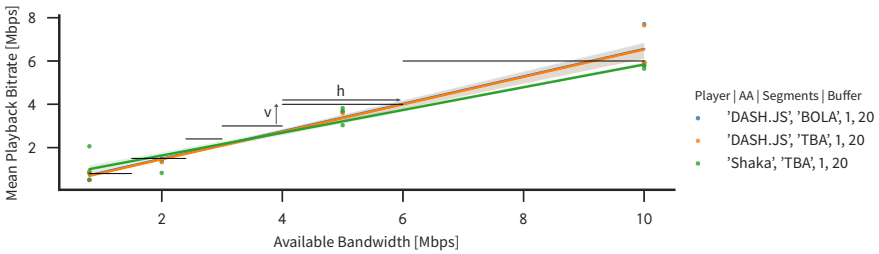
Since higher-quality segments possess larger file sizes we deduce that the time used for fetching segments in the second constellation has a high probability of exceeding the duration of the video content that is currently buffered. Thus, this constellation of small target buffer sizes in combination with relatively long segment lengths can lead to more stalling events, as shown in the previous section.

[21] *This characteristic of re-requesting segments in higher bitrates is, e. g., implemented in YouTube's DASH player, as described by [Mon+17]*

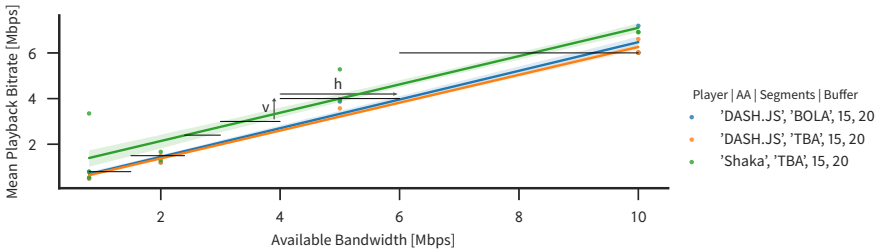
5.3.2.4 *Playback Bitrate*

So far, we considered stalling performance metrics and buffer distributions that indicate how well players mitigate such adverse effects by aligning the requested video qualities with the network and player conditions. These adaptations directly impact the (mean) playback bitrate, an important factor for QoE. Note that the achievable mean playback bitrate depends directly on the available network

bandwidth and the available video representations on the server. Figure 43 provides



(a) All variance environments with 1-second segment length



(b) All variance environments with 15-second segment length

Figure 43: Mean playback bitrates for all available bandwidth environments and 20 seconds buffer sizes with varied bandwidth variance and segment sizes. DASH.JS and the Shaka Player show a consistent behavior. The transparent pipes show a standard 50 percent percentile deviation width. Black stairs show the representations available in the dataset. Vertical differences v at the beginning of the black stairs indicate a QoE degradation, given the networking conditions. Horizontal deviations h indicate efficiency, i. e., how much available bandwidth is required to sustain a quality bitrate.

a comparison of the considered players showing the distributions of average playback bitrates when varying the network bandwidth conditions and the segment lengths. The black stairs show the representations available in the dataset. We use the crossing points of the mean playback bitrates and the representation stairs to express *efficiency* (given as the horizontal deviation h in Figure 43). This indicates the additionally required available bandwidth to sustain a given representation (given stable/volatile networking conditions as depicted). The vertical deviations v of the quality bitrates and the black stairs in Figure 43 denote the following: If the black stairs are higher than the quality bitrate lines, the deviation denotes a loss in QoE in terms of the mean quality bitrate given a certain bandwidth condition. If the black stairs are lower than the quality bitrate lines, then the available bandwidth is sufficient to sustain this representation and the excess available bandwidth is used to occasionally fetch a higher than sustainable bitrate. With Figure 43 we show that the investigated DASH players achieve comparable performance with respect to the mean playback bitrates, given various adaptation algorithms. While variance in the available bandwidth slightly decreases the playback bitrate, increased segment lengths increase the playback bitrate in the

case of the Shaka Player, however not for DASH.JS, irrespective of the AA where no significant difference in the mean bitrate relation is present.

5.3.2.5 *Adaptations*

DASH players employ quality adaptation algorithms to provide an overall better QoE, i. e., by improving metrics such as the average quality bitrate and by avoiding stalling. Adaptation algorithms take specific information as proxy for network and player states and translate this information into decisions on the streaming bitrate, influenced by models implemented in the player as well as by the environment. Various QoE models, as for example discussed in [Seu+15], conclude that the number and the magnitude of adaptation events within a stream is detrimental for QoE. Further, stepwise adaptation is favorable compared to large adaptation steps according to Zink, Schmitt, et al. [ZSS05]. Hence, the streaming performance directly depends on the design of the quality Adaptation Algorithm within the player. Figure 44 shows the distribution of the adaptation amplitudes for different players and AA for all given environment parameters. Here, one surprising observation is that the impact of different adaptation algorithms, e. g., within DASH.JS, is relatively moderate.

From Figure 45, we find that DASH.JS has the highest share of only one adaptation event when using the default buffer configuration.

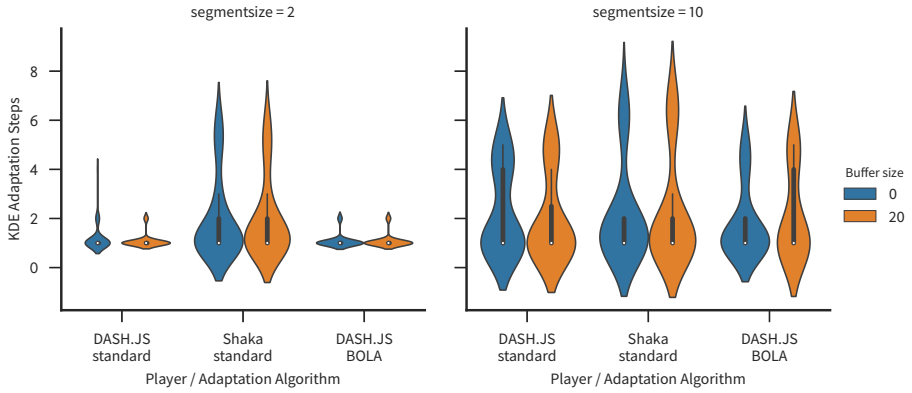


Figure 44: Quality adaptation amplitudes for combinations of players, and AAs for default and 20-second buffer configurations. Respective shapes show the median and distribution. Here, inner black boxes represent median and IQR, with extending lines indicating the full distribution span. Further, the surrounding areas indicate this distribution as a kernel density estimate, with the respective width representing the relative number of observations.

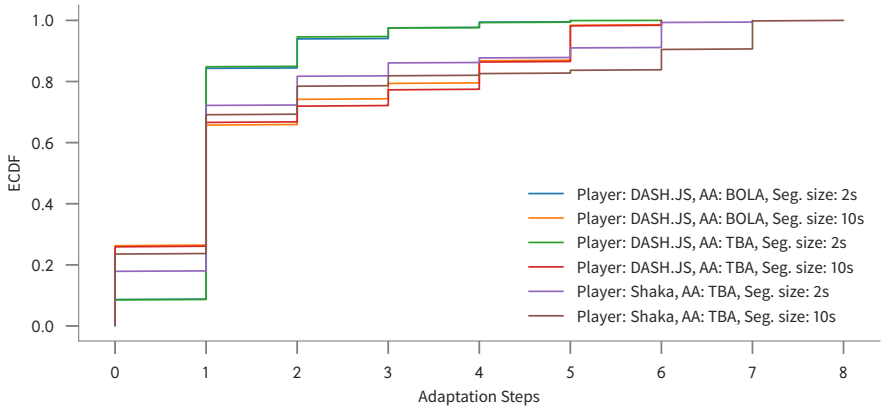


Figure 45: ECDF of adaptation count for combinations of player, AAs and segment sizes

5.3.3 QoE-Centric Analysis and Design Trade-Offs

While QoS attributes give an initial insight in performance attributes of DASH system configurations, understanding the direct impact on users is hard to quantify using this method; QoS metrics, in relation to users, do not generally map linearly to user satisfaction. One concrete example can be given for stalling events and durations: even though a stalling duration of 1 second may be generally preferable to 2 seconds, considering a case where the playback of 60 seconds video is interrupted 10 times compared to a single interruption of 2 seconds may be a far superior experience for users. For example, using the previous example in the model to illustrate this relation for $MOS_{Stal.}$ by Hoßfeld, Seufert, et al. [Hoß+14b], we derive a measure of 2.14 and 3.27 respectively. Exactly such relations, that quantify the users' Quality of Experience (QoE) to a measure called MOS are used in this work. As introduced in Section 2.4, such models are central to the analysis presented in this work and the following evaluations.

However, the one-dimensional view on MOS measures does not always sufficiently quantify QoE, as these measures depend on all other quantities, such as bitrate, to remain unchanged. Thus, one-dimensional MOS metrics need to be considered in relation to one another to quantify their impact on user experience adequately. As maximizing such metrics is, in many cases, correlated with decreasing another metric, i. e., maximizing stalling MOS by reducing total stalling time and events is best achieved by minimizing playback bitrate—and therefore $MOS_{Rep.}$, we analyze such trade-offs in MOS metrics. With the given, illustrative examples, we begin the following analysis on this stalling-based measure, which is a central aspect in the adaptive streaming user experience; later we present the trade-offs involved when increasing such scores in one dimension, against other QoE and QoS metrics related to the playback bitrate.

5.3.3.1 *Stalling Mean Opinion Score*

Figure 46 illustrates the $MOS_{Stal.}$ distribution aggregated over repetitions of the same configuration for common delay and packet loss scenarios in varying TCP

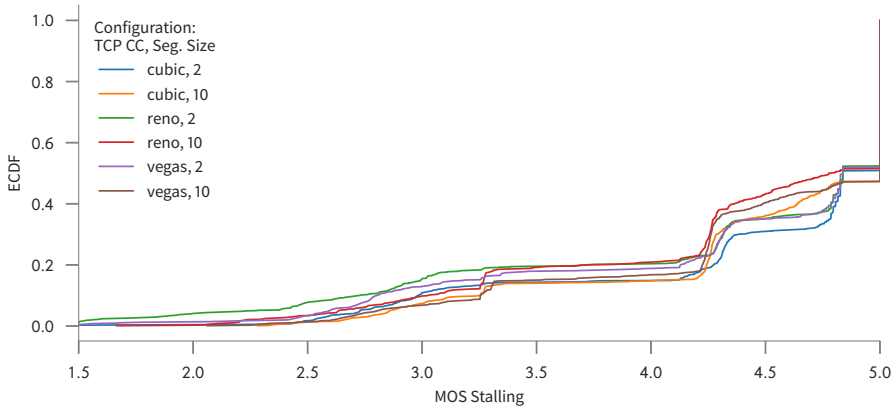


Figure 46: ECDF of $MOS_{Stal.}$ by CC and segment size for Shaka Player using the default TBA.

CC configurations. The range of observed $MOS_{Stal.}$ values indicates significant performance differences depending on the TCP CC algorithm and segment sizes. Here, 10s segment sizes clearly dominate in terms of $MOS_{Stal.}$ across all observations and TCP Cubic provides a performance advantage over TCP Vegas and TCP Reno.

Extending this analysis across players, as shown in Figure 47, illustrates that the choice of the AA is dominated by the choice of the player and its configuration with regard to the segment size. For DASH.JS, the distribution of $MOS_{Stal.}$ is dominated by the selected segment size, while the AA plays only a minor role in the performance difference with regard to this metric. When comparing the same class of TBA AA between players, however, the Shaka Player's performance relates directly to the segment size, where the best overall distribution on $MOS_{Stal.}$ is achieved with 10s segments.

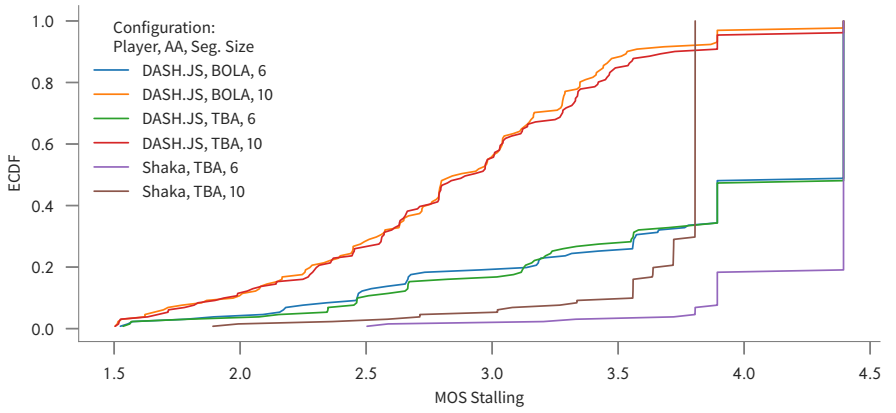


Figure 47: CDF of $MOS_{Stal.}$ by player, AA and segment size.

5.3.3.2 *Stalling vs. Bitrate Mean Opinion Score*

To evaluate the impact of aggregated configurations, Figure 48 presents different configurations and their QoE metrics along the axis for $MOS_{Rep.}$and $MOS_{Stal.}$. The QoE difference between the Pareto optimal configuration and non-Pareto

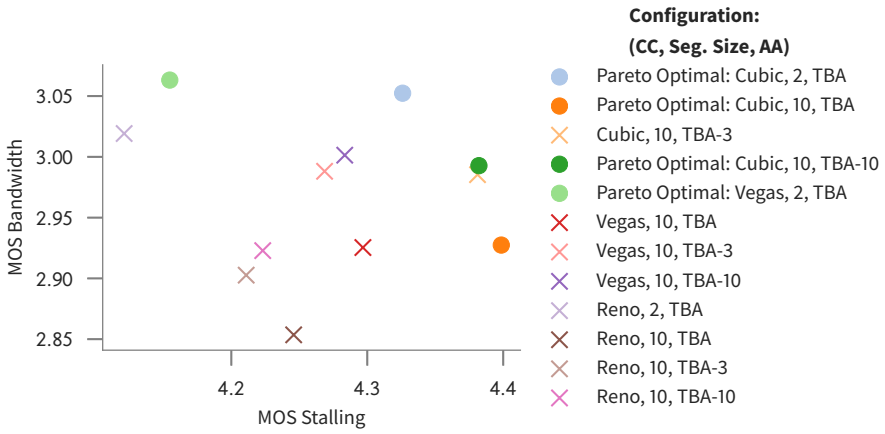


Figure 48: The QoE metrics for all combinations of bandwidth estimators, segment sizes, and congestion controls, compared with an adaptive solution which uses the best mechanism combination per network condition.

optimal configurations shows a large performance improvement potential by selecting configurations including CC, segment size and TBA attributes in the Shaka Player. These, however, depend on the concrete trade-off between these opposing optimization goals. Here, we observe that for all but one Pareto optimal configurations, TCP Cubic is part of the configurations. We also observe that, in terms of segment size, larger segments increase $MOS_{Stal.}$while smaller segments provide a better $MOS_{Rep.}$.

5.3.3.3 *Playback Bitrate vs. Stalling Quality of Experience:*

We analyze the trade-offs between two previously discussed aspects, *(i)* the playback bitrate and *(ii)* stalling, captured as $MOS_{Stal.}$. Figure 49 shows a scatter plot of $MOS_{Stal.}$(the higher the better) and the achieved mean playback bitrate, where each entry in the graph denotes the performance of a configuration averaged over all considered network conditions. Interestingly, the depicted Pareto frontier shows that no single player (configuration) dominates both metrics. In particular, all players and AAs are represented at least once on the Pareto frontier. Thus, for every player and, AA there exists a sweet spot and a weighted aggregate taking stalling QoE and mean playback bitrate that shows that this configuration is superior.

The Pareto frontier further shows that large buffer sizes dominate the performance for all player configurations for both considered metrics. We also note that moving to higher playback bitrates on the Pareto frontier corresponds to increasing the segment length of the corresponding configurations. It is also evident that combinations of small buffer sizes and long segments perform badly.

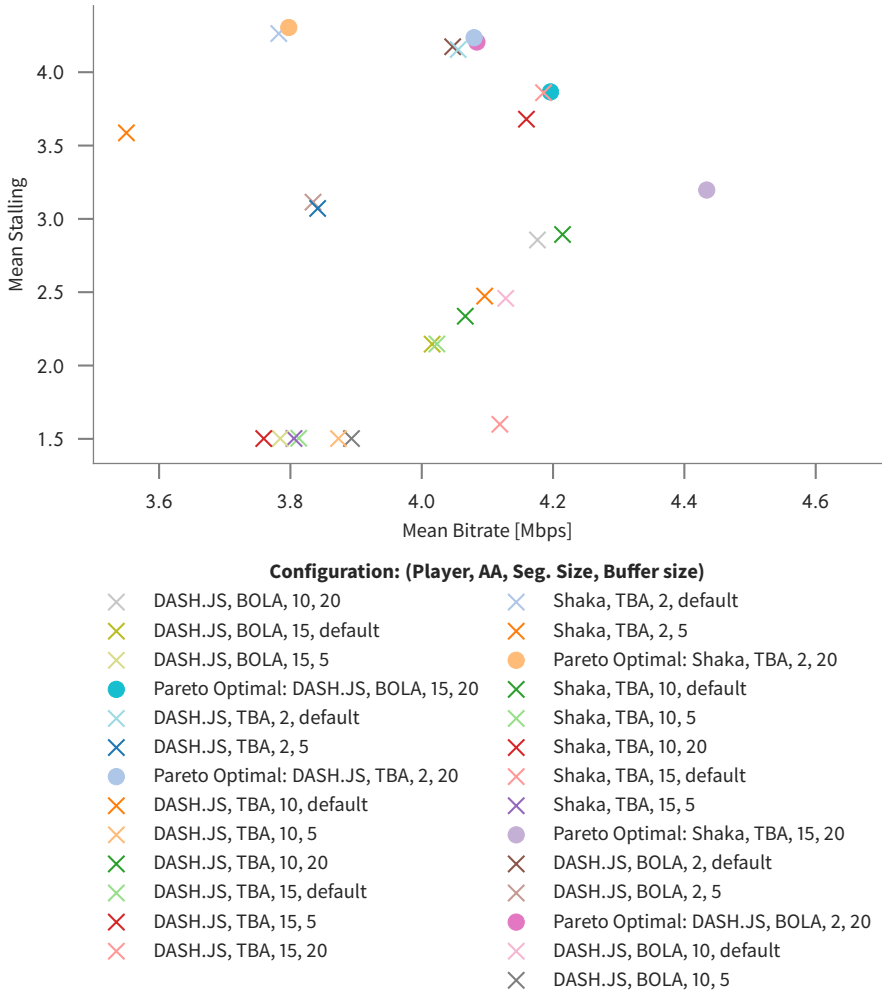


Figure 49: Trade-off between playback bitrate and stalling QoE for different configurations, aggregated over all analyzed environment conditions.

Last, Figure 49 shows the minor impact of different adaptation algorithms within DASH.JS.

5.3.3.4 *Adaptations vs. Stalling Quality of Experience:*

As quality adaptations are used to avoid stalling, we analyze the trade-off between the number of adaptations and the stalling QoE. Figure 50 shows the average stalling QoE (the higher the better) and the average number of quality bitrate adaptations for different player configurations in various networking environments. Here too, we note that all players and adaptation algorithms are represented on the Pareto frontier such that no single player configuration dominates. The adaptation algorithm choice within DASH.JS shows again nearly no impact.

The figure shows that by allowing a few adaptations a substantial increase in stalling QoE is achieved. We note that the behavior of both players differs

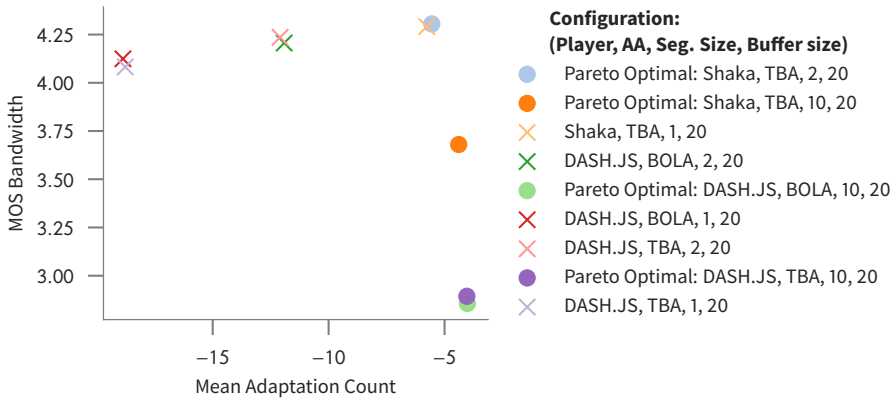


Figure 50: Trade-off between adaptation count and stalling QoE for different configurations, aggregated for all analyzed environment conditions.

with regard to segment length. For DASH.JS, the adaptation count significantly increases for smaller segment lengths. In contrast, Shaka Player does not show such a dependency.

5.3.4 On Improving QoE Trough Transitions

In this section, we present an evaluation on learning classification models for selecting *sweet spot* configurations (i. e., mechanisms combinations) depending on environment parameters, i. e., environment conditions in which players with specific cross-layer DASH configurations provide the best performance.

The goal of this evaluation is to motivate the selection of such mechanisms (concrete combinations of configurations in the DASH design space) that allow for transitions based on measured environment conditions and ultimately improve QoE by a context-dependent reconfiguration during runtime.

While there are undoubtedly technical challenges involved with transitioning between such configuration states—that involve reconfiguration on multiple layers—there are examples in the different domains that motivate the feasibility of this concept. For instance, *i)* adjusting buffer sizes is dynamically supported by DASH.JS's and Shaka Player's API; *ii)* regarding the AA, Spiteri [Spi18] introduces a practical example of dynamic switching between DASH.JS's Adaptation Algorithms, TBA and BOLA during runtime, *iii)* varying request ranges and thus, adjusting the relevant size of segments to be requested is already actively employed in commercial players, i. e., by YouTube as introduced by Mondal et al. [Mon+17]; *iv)* transparent switching between TCP flows could be achieved based on Multipath-TCP (MPTCP) (also see Frömmgen, Rizk, et al. [Frö+17]); *v)* and last, general models for mechanism transitions have been proposed and demonstrated in [Ric17] or by Frömmgen, Rizk, et al. [Frö+17].

The following configurations show, mostly for the sake of representation, an environment condition limited to two dimensions. Here, beginning with varying degrees of bandwidth variance and means, we identify the *best* given configuration along the axis of the presented conditions. We then explore this learned relation

of configurations to the environment space of loss and delay. We also include CC as a configuration option.

The underlying classification method is based on a simple TreeBased learning model to generate rules that could be deployed to clients, thus allowing for a potential client-side state migration when transferring these models (e. g., along with the MPD). Generally, such learning-based approaches can be extended to a

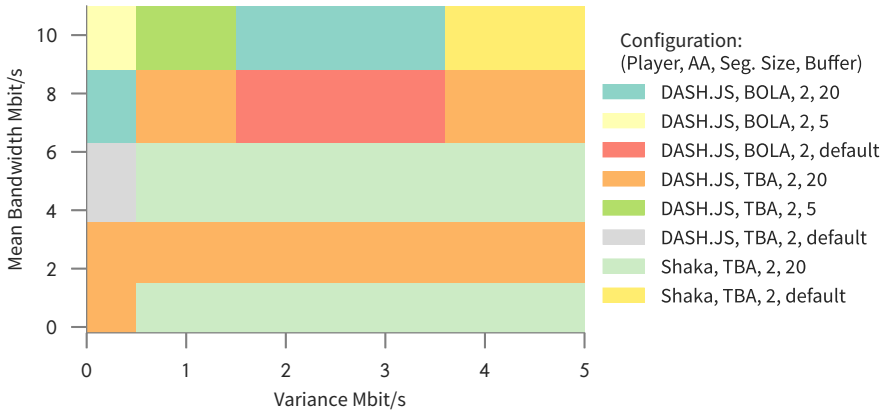


Figure 51: Best configuration for $MOS_{Stal.}$ using a decision tree learning approach.

multi-dimensional environment space, and other suitable learning approaches may be employed, such as Neural Networks.

In Figure 51 we present such a classification (based on dataset 1) depending on the bandwidth conditions, varied by their mean and variance. The first observation is that, here, all players and AAs buffer configurations are present. This observation aligns with the previous analysis made in this chapter, where we presented evidence for performance differences in $MOS_{Stal.}$ beyond the selection of the AA. Only regarding the segment size, we see that for the target measure $MOS_{Stal.}$ a 2-second

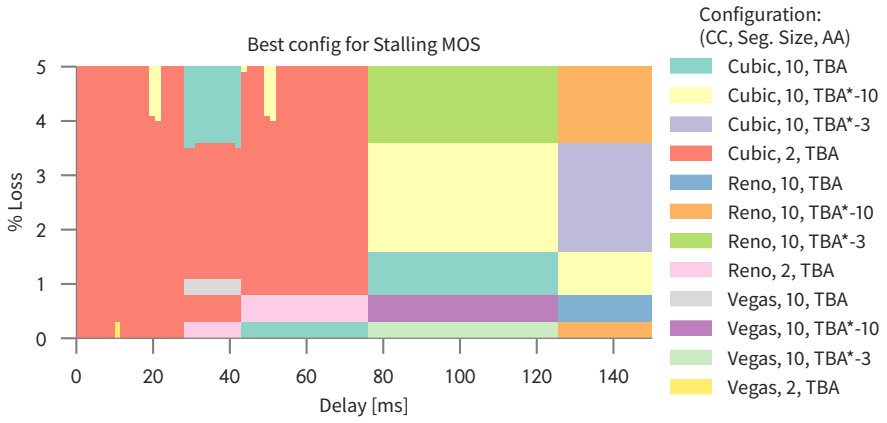


Figure 52: Best $MOS_{Stal.}$ using a decision tree learning approach for a dynamic bandwidth environment with adjusted delay and loss.

segment configuration is always selected. Overall, we see that configurations with large buffer sizes dominate for low bandwidth, high variances configurations. Surprisingly, for high bandwidths above 6 Mbps, also low buffer size configurations are selected, mostly using the BOLA AA.

In the second analysis, shown in Figure 52 based on dataset 2 including TCP CC using a single player (Shaka Player), there is, analog to the previous analysis, a clear dominance of the Cubic CC when learning for the best configurations on $MOS_{Stal.}$, especially with regard to low delays smaller than 80ms.

However, in contrast to the previous analysis, we also can observe the selection of larger segment sizes of 10 seconds, especially in high delay environments. Notably, the learned configuration space, again, spans all configuration variables, including the adjusted TBAs, CCs and segment sizes.

Last, for the same environment space, Figure 53 shows a more mixed landscape of potential configurations, where both, Cubic and Vegas CCs are present across the entire configuration space.

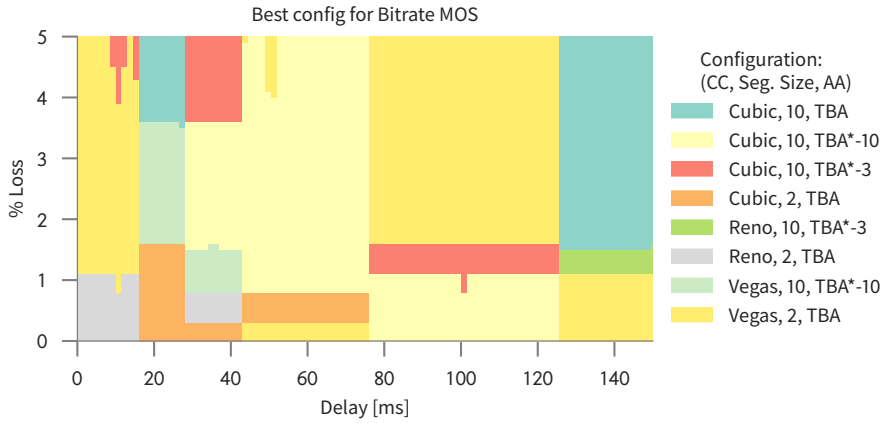


Figure 53: Best $MOS_{Rep.}$ using a decision tree learning approach for a dynamic bandwidth environment with adjusted delay and loss.

Given appropriate models that combine such opposing QoE optimization goals as $MOS_{Stal.}$ $MOS_{Rep.}$, these analyses can be extended to derive configurations to maximize the overall QoE. Such models are, at the time of writing, subject of active research see, e. g., [Dua+17; GPB18; Hoß+17].

5.4 *Conclusions*

In this section, we provided a systematic study of the impact of the DASH player choice and the cross-layer configuration space on the streaming performance.

Here, we first motivate a concept for generating an extensive set of evaluations by establishing an execution environment to reproducibly monitor and evaluate the performance of real-world and academic DASH players. Further, with this work, we demonstrate the capability of the general network experimentation framework *maci* as the underlying approach for our empirical DASH evaluation that allows reproducible comparisons of such real-world DASH players in emulated

networking environments. Two datasets, derived with this extensive evaluation approach, are representing the focus on TCP CC and DASH players, respectively, as the foundation for our evaluation.

In particular, our results show that *i)* player performance affinities show that suitably configured players can be superior with respect to given QoE performance metrics; *ii)* none of the existing DASH configurations outperforms all other configurations, however, a set of Pareto optimal configurations can be identified; *iii)* the choice of the target buffer size together with the player implementation, in most cases, dominates over the influence of AAs with regard to MOS measures; *iv)* larger segment sizes are generally favorable for improving the MOS, in particular in combination with TCP Cubic.

This stands in contrast to the majority of research efforts that are being directed towards investigating improvements in AAs that relies on a small set of either environment or configuration parameters. Further, our developed methodology allows an informed player selection and configuration at the beginning of streaming sessions to maximize QoE, as well as evidence for transitions between identified configurations during runtime.

6 Video Streaming in Future Networks

With the success of Video on Demand (VoD) and Mobile Video Broadcasting Services (MBSs) on the Internet, there is an ongoing rise of demand for data transmission in networks. Traditionally these technical demands are being solved by employing a Content Delivery Network (CDN) and by scaling up resources.

Nonetheless, rising costs, capped bandwidths, and higher video quality (i.e., UHD, HDR), together with the users' more demanding service quality expectations, make effective bandwidth use and mobility support crucial considerations in the design of video streaming systems.

In this context, a set of networking concepts qualify to deliver such efficiency improvements for video streaming: Peer-to-Peer and multicast-based approaches, for example, explored in Rückert [Rüc16]. While these approaches can often be readily integrated into existing infrastructures, their intended use is within current networking concepts, relying on established routing and addressing schemes. A different approach is considering cross-layer modifications, i. e., examining the potentials for video delivery considering innovative networking paradigms that influence the entire network stack. While this comes with the requirement of more fundamental changes to current infrastructures, it provides the opportunity to tap into a higher potential.

To this end, the last chapter of this dissertation addresses such potentials in the domain of Information-centric Networks (ICNs). This concept allows, for example, service providers to directly address very dynamically created video content coming from and being requested by mobile users while still allowing for a flexible and scalable networking architecture with support for caching. Further, this implicit caching support, in conjunction with content-centric addressing, can increase efficiency in supplying frequently accessed content, by resolving requests as close as possible to a given consumer. As part of this dissertation, we investigate the performance of Dynamic Adaptive Streaming over HTTP (DASH), in particular, the design of Adaptation Algorithms, in conjunction with Named Data Networking (NDN)-based delivery. Here, we focus on the influence of the mentioned caching concept and show the need to reconsider DASH Adaptation Algorithm concepts by analyzing and proposing extensions to a state-of-the-art approach, for such future use cases, as presented in [Sto+18]

In conjunction with this work, we further highlight the potentials in this domain by presenting a case-study on the technical feasibility of using DASH streaming over NDN in a physical testbed, based on our publication in [Sto+16a].

6.1 *Motivation for NDN-based DASH Delivery*

The present Internet architecture is based predominantly on serving static content related to IP-addresses identifying concrete hosts. This host-centric design, however, has limited potential when trying to address dynamically created resources, such as present in Mobile Video Composition (MVC), and the discovery of content, based on DNS, is limited in its capability to address content independent of concrete hosts.

These limitations motivate us to reconsider the underlying networking paradigm, in particular, ICN, as a promising candidate to address these limitations [Ahl+12]. Concerning DASH, the underlying idea of using this concept is that users are interested in the content (i. e., the video segments) rather than any particular copy or location. Hence, the *content* itself is made a primary object of network search, transfer, addressing, and retrieval.

Separating content requests from the explicit knowledge of host addresses and shifting the responsibility of content discovery to the network brings many advantages for DASH, such as: *i)* more robust transitions between networks during streaming sessions [Sto+16a], *ii)* support for caching at each forwarding entity in the network to fulfil the demand more efficiently when compared to dedicated CDN nodes[FZX16], *iii)* network coding to efficiently combine replies by multiple content stores [RWS17], *iv)* and the ability to implement content-aware forwarding and caching mechanisms to improve the overall efficiency of the network. Yet, with such fundamental changes in the underlying networking concepts, new challenges arise for existing services such as DASH since they rely directly on a host-bound network architecture. In particular, DASH adaptation depends on the measurement history between a particular host and client, and thus, assumes host-bound connections. Since this is not provided in NDNs it can lead to problematic side effects when using DASH in an NDN context. Most crucially, stalling may occur when segments are not available in the cache [Liu+13].

In this study, we provide a systematic evaluation of the impact of an NDN system on standard DASH Adaptation Algorithms. To this end *i)* we propose an NDN evaluation architecture making use of established emulation concepts within `Mininet`[bi], *ii)* we show that implicit caching in NDN leads to undesirable adaption effects using traditional DASH adaptation, *iii)* and we propose a set of extensions for using *any* throughput-based AA in NDN providing an overall improved Quality of Experience (QoE) based on extended segment information in Media Presentation Descriptions (MPDs) and NDN-based throughput measurements on NDN chunk granularity.

We begin with an outline of our approach used for the emulative evaluation of DASH in NDN, where we introduce a novel concept relying on container-based evaluation in `Mininet`. This is followed by a preliminary study on the influence of caching in DASH adaptation. Given the concrete motivation based on these observations, we propose a new concept for DASH Adaptation Algorithms in NDN on the concept of chunk-based measurements. Last, we evaluate the

[bi] http://mininet.org/

proposed concept in comparison to state-of-the-art DASH Adaptation Algorithms BOLA [SUS16] and PANDA [Li+14].

6.2 *Emulative Evaluation of DASH in Named Data Networks*

In the following, we give an overview of our emulation environment used for the evaluations presented in Sections 6.3 to 6.5, along the lines of cross-layer environment configurations (see Table 13).

6.2.1 *Network Environment*

Goal is to evaluate NDN-based DASH, mimicking realistic conditions as closely as possible while maintaining reproducibility of the results. Thus, the emulation setup needs to fulfill the following functional requirements: *i)* support for large, dynamic topologies, *ii)* within an emulated network environment, *iii)* based on the original NDN source code. To this end, we built an NDN architecture based on an extension of the `Mininet` project called `containernet`[bj] that allows to execute Docker hosts as nodes in the `Mininet` emulator.

By using `containernet` with encapsulated NDN functionality in Docker containers. We claim that more realistic emulation scenarios can be achieved in this way.

6.2.1.1 *Topology*

The setup used throughout this chapter is depicted in Figure 54. Each of the functional NDN components is represented in separate parts of the network, achieving variations in caching, bandwidth and latency for the clients. The architecture varies in its physical setup, which is similar to an ISP topology, and the logical NDN architecture. From a client's perspective, a flat hierarchy between caches exists, and caches can either retrieve content from other caches or from the server.

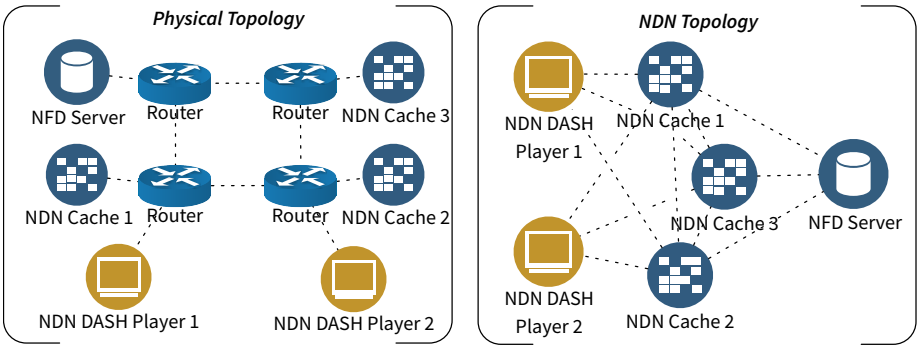


Figure 54: Evaluation topology using depicting NDN software components; NFD is the Networking Forwarding Deamon, NDN Server hosts DASH segments using `repo-ng`.

[bj] https://github.com/containernet/containernet

Table 13: Overview of the different parameters in every part of the system.

Zone	Component	Configuration	Instantiations (this work)	Related Work
DASH Mechanisms	Video Content	Segment length [s]	2 sec	
	DASH Player	Adaptation Algorithm	BOLA, Panda, Panda (extended), AlwaysLowest	[Sam+17]
Network Mechanisms	Client	Congestion Control	ICP	[Sam+17; SCP13]
	Server	Chunk Size	1400 Bytes	[Awi+13]
	Cache	[Number / Capacity / Strategy]	$\{0, 3\}$ / $\{\infty, 1500\}$ / LRU	[Bha+15; GSW13; Liu+13]
	all	Forwarding Strategy	ncc	
Network Environment	Containernet	Bandwidth	dynamic[DAS17; Rii+13], static	[Liu+13]
		Topology	4×OpenVSwitch, 6×Docker Host	

6.2.1.2 *Bandwidth Traces*

To verify the performance of the presented approaches analog to a real-world setting, the system is investigated using traces collected from mobile devices [Rii+13]. We replay these by changing the traffic on the clients' link using `TC`[bk] accordingly. Whereas these dynamic bandwidth traces allow to analyze the impact in conditions very close to reality, we also use static and designed bandwidth profiles for specific testing scenarios.

6.2.2 NDN Mechanisms

In the emulation setup, we distinguish between three types of hosts: *NDN Servers*, *NDN Caches*, and *NDN Clients*. Each type is implemented in a respective container image based on Ubuntu 14.04. All container types run an instance of `NFD`[bl], while caches and the servers use `repo-ng`[bm] to host the video dataset.

6.2.2.1 *Chunk Size*

A core parameter of every content-centric network is the size of data that travels in a single chunk. While the default setting is 1000 Bytes, we implemented a chunk size of 1400 Bytes to allow an easy comparison with related work [Sam+17].

6.2.2.2 *Congestion Control*

Presently, there is no built-in or standard congestion control for NDN, but research is leading into the direction of adding a congestion control protocol as it

[bk] http://lartc.org/manpages/tc.txt
[bl] https://github.com/named-data/nfd
[bm] https://github.com/named-data/repo-ng

is inevitable in networks that operate at full capacity. In this work, we use ICP [CGM12; Ren+15], an Interest Control Protocol which realizes a window-based flow control while aiming for efficiency and fairness. Other approaches such as SAID [Che+16] or CCTCP [SCP13] exist but are not directly usable in our NDN infrastructure.

6.2.2.3 *Forwarding Strategy*

Similar to the congestion control, the forwarding strategy in NDN is replaceable. There are many designs and ideas to improve the forwarding for better efficiency and general usability. In the presented studies, the `access` strategy is configured. It supports a faster retransmission of lost interest messages when compared to the (default) `ncc` strategy.

6.2.3 DASH Mechanisms

The DASH player deployed on the *NDN Client* is implemented with `libdash`, based on the code provided by [Sam+17]. As for the video dataset, a subset of nine layers of the *Big Buck Bunny*[bn] dataset encoded in two-second H.264-AVC segments is used, encoded in bitrates between 1.03 - 4.21 Mbps. Each segment is, when loaded in the NDN data store, uniquely identifiable by name and can thus be stored in caches on the route. It is, however, important to note that datasets with incompatible encoding should be stored in distinct namespaces, even though the video content itself may be identical, as these dependencies cannot be directly interpreted by NDN.

Table 14: Big Buck Bunny DASH dataset representations

Resolution	Bitrate (Mbps)	FPS	Layer Number
1920×1080	4.21	24	9
1920×1080	3.84	24	8
1920×1080	3.52	24	7
1920×1080	3.07	24	6
1920×1080	2.48	24	5
1920×1080	2.13	24	4
1280×720	1.54	24	3
1280×720	1.24	24	2
1280×720	1.03	24	1

6.3 Challenges for DASH over NDN

In the following, we present key challenges for the design of DASH AAs for NDN. In particular, we investigate the interaction between NDN's multi-sourcing and caching capabilities and the throughput measurements used in rate-based DASH AAs.

[bn] http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/bunny_2s/bunny_2s_1200kbit/

To demonstrate the interaction of Throughput-based Adaptation (TBA) and NDN, we use the following scenario: given that two clients play the same video in sequential order while using the `AlwaysLowest` AA, i. e., continuously playing the lowest quality, we know that the second client has relevant video segments located in the caches when the playback is initiated. In this setup, which mimics Figure 54, the server link bandwidth is set to 4 Mbps, and the cache link bandwidths are given by 10 Mbps (`Cache 1`), 20 Mbps (`Cache 2`), 30 Mbps (`Cache 3`). In Figure 55 we show the Empirical Cumulative Density Function (ECDF) of the measured throughput on an NDN chunk basis. All requests from the `Client 1` (yellow) are served by the server as the caches are empty. Our first observation here is that although the link bandwidth to the server is fixed to 4 Mbps some chunks are received with a much lower throughput which we attribute to the ICP congestion control in NDN. Note that this throughput measurement, i. e., on NDN chunk basis, is the foundation for the input to rate-based DASH algorithms. For `Client 2` about half of the requests are fulfilled with a bandwidth of 10 Mbps while the rest is widely distributed. Again, we observe NDN chunk throughputs at both ends of the scale which we attribute to congestion control. Intermediate bandwidth clusters are caused by ICPs Congestion Control (CC) mechanisms.

Note that on a DASH segment basis, i. e., aggregating hundreds of NDN chunks into one DASH segment, the altered by these outliers. Hence, the existence of caches causes the client to observe different effective data rates (see Figure 55) which introduces fluctuations between the requests.

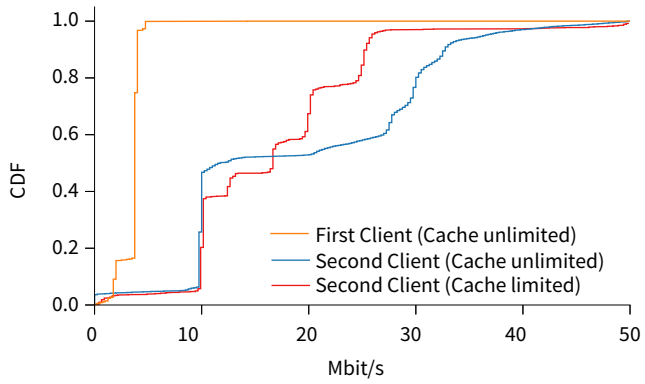


Figure 55: Two clients playing in sequential order using an `AlwaysLowest` AA. The server link bandwidth is given by 4 Mbps, while the caches' link bandwidths are 10 Mbps (`Cache 1`), 20 Mbps (`Cache 2`), 30 Mbps (`Cache 3`), respectively. The influence of caching becomes evident by the higher bitrate for the second clients. The cache size for the second client (red) was limited to 5×10^3 NDN chunks.

Overall, we observe that the key point to be addressed for DASH in NDN are the large variations in throughput measurements due to caching which leads to a wrong interpretation of throughput estimates with common Adaptation Algorithms [Liu+13].

Table 15: Comparison between the standard PANDA algorithm and the proposed extensions to PANDA represented by adaptation and video quality metrics. The used test cases did not exhibit any stalling.

Algorithm	PANDA (a)	PANDA (b)	PANDA (c)
Adaptations [#]	80	58	29
Magnitude [level]	0.63	0.38	0.18
Mean-based Bitrate	2.66	2.85	2.42
Segment-level Bitrate	2.31	2.57	2.20

6.4 Approach for Improved DASH Adaptation in NDNs

To address the analyzed challenges, we propose three extensions for throughput-based DASH AAs in NDN based on the example of PANDA. However, as the discussed extensions are generic, they can also be used with other throughput-based adaptation approaches.

6.4.1 Chunk-based Measurements

The design of NDN allows streaming applications to obtain additional information about NDN's chunk downloads, e. g., the chunk-based throughput. For standard DASH over TCP/IP the client receives segment-based or byte range-based throughput estimates of the connection to the server. In contrast, in NDN, estimates can be obtained for both, DASH segments and (in a finer granularity) NDN chunks. Hence, existing DASH Adaptation Algorithms may be adjusted or new approaches designed to take advantage of this information.

Since the data packets in NDNs are usually small (a few kilobytes) compared to the video segments (up to many megabytes), the number of measurements for each requested segment is very large. Single NDN chunks are—because of their size—prone to variations when being transmitted through the network as even small variations in transmission time can have a high impact on the throughput estimates. This implies that a high measurement accuracy is required which is also important for increasing the robustness of the feedback to the adaptation algorithm. For example, the correct assignment of Interests being sent off and their matching Data packets coming in at the client is important to gather the correct timing values for throughput estimations in rate-based or hybrid adaptation algorithms. Other considerations include the pipelining of Interests (sending several Interests in close succession) and possible out-of-order reception of Data packets.

For the throughput measurement based on NDN chunks, we use an approach that has been used for measurements of TCP/IP connection throughput introduced by Khangura et al. [KF17]. The authors suggest a bandwidth estimation from passive measurements based on the *ACK-GAP* model. Note that the TCP sender-side model with ACKs is the inversion of the NDN client-side model: In NDN, the client handles the content flow by sending Interest packets to the server, and the server answers with the sending of Data packets. The Data packet is received by the client, and the rate estimation is done. This is the same problem as in TCP,

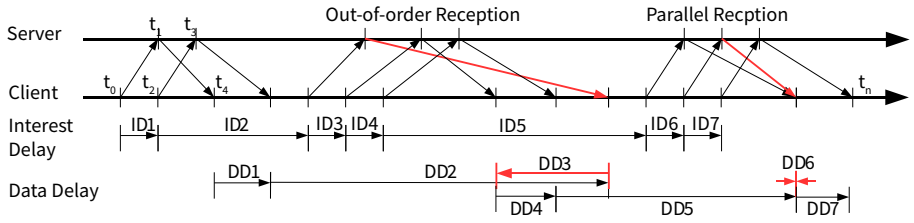


Figure 56: The three different scenarios for Data packet receptions.

where only the server and the client change positions, and the server starts with the data transfer, the client answering with an ACK.

Since the measurement is conducted over the entire downloading session, no extra probing packets are needed; feedback is extracted constantly with data transfer. However, post-processing of the bandwidth estimations is needed in all cases. Because the measurements are based on the delay of received Data packets that were requested consecutively, we can only estimate the throughput from the second received chunk on, but since the number of feedbacks for the chunk-based granularity is large, i.e., about 2-4 orders of magnitude higher compared to the segment-based granularity in traditional networks, this loss is negligible. Additionally, not all Data packet receptions provide usable timing values when special scenarios occur. The three different scenarios of Data packet receptions are depicted in Figure 56. In the common case, the Data packets are received in the same order as the Interests were sent off, and every data delay provides a valid timing value. When an out-of-order reception happens, the delay between two received Data packets is negative ($DD_3 < 0$). The third case is the parallel reception of two Data packets. Here, the data delay equals zero ($DD_6 = 0$). Out-of-order and parallel receptions lead to invalid computation outputs because a not strictly positive timing value is not usable for rate estimations. Thus, the computations based on these two failure scenarios are discarded.

Based on our preliminary investigation, we found that a sample should contain about 50 fine-grained computations for providing a very frequent–but reliable–feedback. While a higher sample size up to 100 is still reasonable, samples using less than 50 computation values suffer more from single measurement failures. It is important to note that the sample size corresponds to the chunk size in NDNs: when larger NDN chunks are used, the measurement failures become less likely. Currently, we use the harmonic mean as a method for sampling.

By considering feedback using more samples to represent the current throughput, we improve the information gain for AA. We argue that in more complex NDN scenarios (e. g., involving multiple caching entities), it is important to get the direct network feedback instead of relying on indirect feedback (as in DASH buffer-based measurements). The main advantage we advocate is a faster detection of source changes within single requests of DASH segments that can be used within the decision-making process of AA. Using Panda as an example, Table 15 shows the resulting adaptation behavior. While an overall high playback bit rate can be achieved, the number of adaptation steps is negatively affected.

6.4.2 Extended MPD

While the above method of throughput measurement is a NDN-specific approach, a further general approach to provide a better Adaptation Algorithm performance and thus QoE is to use more precise information before determining an adaptation. Currently, the decision on the quality level chosen by a rate-based Adaptation Algorithms for the next segment download is based on the `bps` value of the different quality levels obtained from the MPD. Usually, this value is the arithmetic average over the segment rates (segment size divided by segment length) that represent a specific quality level. A problem here is the high degree of variance in segment sizes rates around the mean bitrate, as discussed in Subsection 2.4.1 (see Figure 9).

Thus, in previous work, Juluri et al. [JTM15] and C. Wang et al. [WRZ16] propose to include the segment size in the MPD which is adopted in our discussed Adaptation Algorithm concept.

6.4.3 Adaptation Hysteria Reduction

To address increased adaptations due to throughput fluctuations, we propose two extensions for TBA algorithms in order to utilize the advantages of the more precise network measurements as well as the segment size knowledge.

6.4.3.1 *Instability*

To quantify the instability during playback, we employ a measure by [JSZ14; Li+14]. Let r_t be the mean-based quality layer bitrate fetched at time t. The instability of a playback at time t is defined in the work above as:

$$I_t = \frac{\sum_{d=0}^{k-1} |r_{t-d} - r_{t-d-1}| w(d)}{\sum_{d=0}^{k-1} r_{t-d} w(d)} \tag{2}$$

using $w(d) = k - d$. This instability formulation puts a stronger weight on recent adaptations to detect fluctuations, hence, $I_t \geq 0$. For no fluctuations, I_t is zero. We use $k = 10$ to increase the weight of more recent segments. In subsequent adaptation decisions, if $I_t > 0.05$ an adaptation to higher qualities is postponed to reduce short-term fluctuations.

6.4.3.2 *Local Segment Rate Variation*

To address the rate sustainability of adaptation decisions we use the following algorithm that we denoted as local segment rate variation function. This time, we use R_t to denote the segment-level quality bit rate of the specific video segment at time t (see Subsection 6.4.2) and B_t, which is the currently measured bandwidth, as described in Subsection 6.4.1. For each triggered adaptation to a higher quality level, the following decision process is executed:

If `skipAdaptation` is set to `true`, a planned adaptation to a higher quality is skipped. This extension is different from standard rate adaptation comparisons as it takes the *local variation* of the segment sizes into account. Therefore, adaptations to higher quality are only permitted if they can be sustained over a longer period of time and not just for a single video segment. We recommend a window size of

Algorithm 1: Process to determine if a planned bitrate change can be supported for the next i segments

```
window: 3;
skipAdaptation: False;
for 1 ≤ i ≤ window do
    if B_t ≤ R_{t+i} then
        skipAdaptation = True;
```

3 segments because a bigger window would impede the adaptation algorithm too much while a smaller window would not stabilize the video quality well enough.

For evaluating the impact of our extensions, we used DASH-IF test cases [DAS17]. Table 15 shows the standard PANDA algorithm, the PANDA algorithm which uses the instability function and the PANDA algorithm extended with the instability function as well as the local segment rate variation function. Overall, only the third PANDA configuration can provide a switching magnitude that is low enough for a good QoE while still providing a high bitrate. When PANDA is extended with the instability function exclusively, there are still too many switches which last only for one single segment. This lowers the QoE more than a higher bitrate can improve it.

6.5 *Evaluation of Chunk-Aware DASH Adaptation in NDN*

To evaluate the performance of the discussed DASH AA we are using a trace-based emulation of the client bandwidth as described in Section 6.2. The server link capacity is 4 Mbps, and the Cache link capacities are given by 10 Mbps (`Cache 1`), 20 Mbps (`Cache 2`), and 30 Mbps (Cache 3), respectively. Playback is initiated sequentially for two clients, i. e., the second client encounters a situation where content is already cached. We test the three Adaptation Algorithms separately, i. e., *BOLA*, *Panda* and the *extended version of Panda* that is proposed in this work. To ensure a fair comparison between these algorithms, a maximum playout buffer size of 20 seconds is configured in each case.

Investigating the respective first clients in Table 16 we observe that the stalling time is shorter compared to the second clients once content has been cached. This effect is particularly pronounced in the regular version of Panda. The extended version of Panda, as proposed in our work, mitigates this effect, resulting in no stalling for both, the first and second client.

While both Panda-based AAs achieve an increase in the playback bitrate compared to the first client, this is not the case for the buffer-based AA BOLA. Here, in case of the second client, the higher available bandwidth, available by caches populated with previously requested segments, is not sufficiently utilized. This results in a lower average bitrate for the second client.

Overall, the extended version of Panda shows a stable playback behavior, which is comparable to BOLA in terms of adaptation steps for the first clients. While in the extended Panda AA's case, the second client shows more adaptation steps compared to BOLA, it maintains a high bitrate without stalling. Thus, it can be

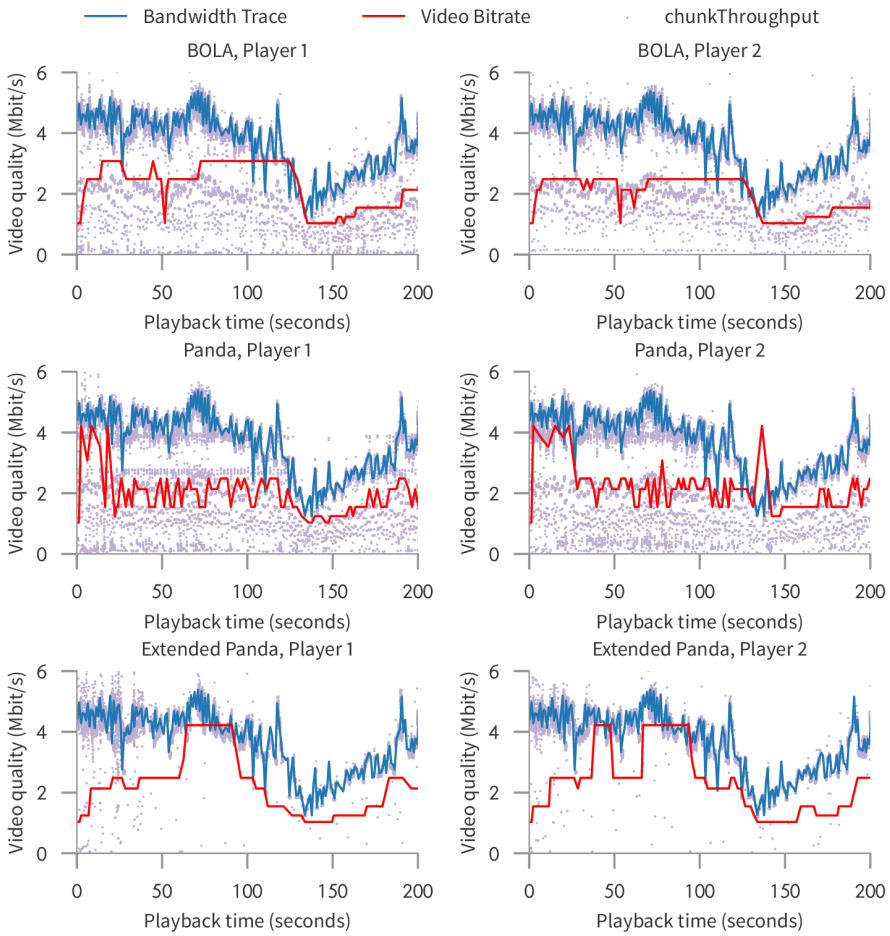


Figure 57: BOLA, Panda, and Extended Panda clients playing, respectively, in sequential order, i. e., `Client 1` followed by `Client 2` for each Adaptation Algorithm, applying the same bandwidth trace for both clients.

argued that the negative effect of a slightly increased number of adaption steps is counterbalanced by the higher bitrate with no stalling. We expect that the fact that no stallings occur at a higher bitrate leads to a higher QoE that can be achieved within a multi-cache NDN scenario when using our extended version of Panda.

The NDN chunk throughput shown in Figure 57 for the extended version of Panda indicates that obtained chunk-based bandwidth samples follow the bandwidth trace for most measurements. This observation is also confirmed when compared with the distribution of these measurements depicted in Figure 58, where roughly 80% of the throughput traces' distribution is closely resembled by measurements obtained in the extended version of Panda.

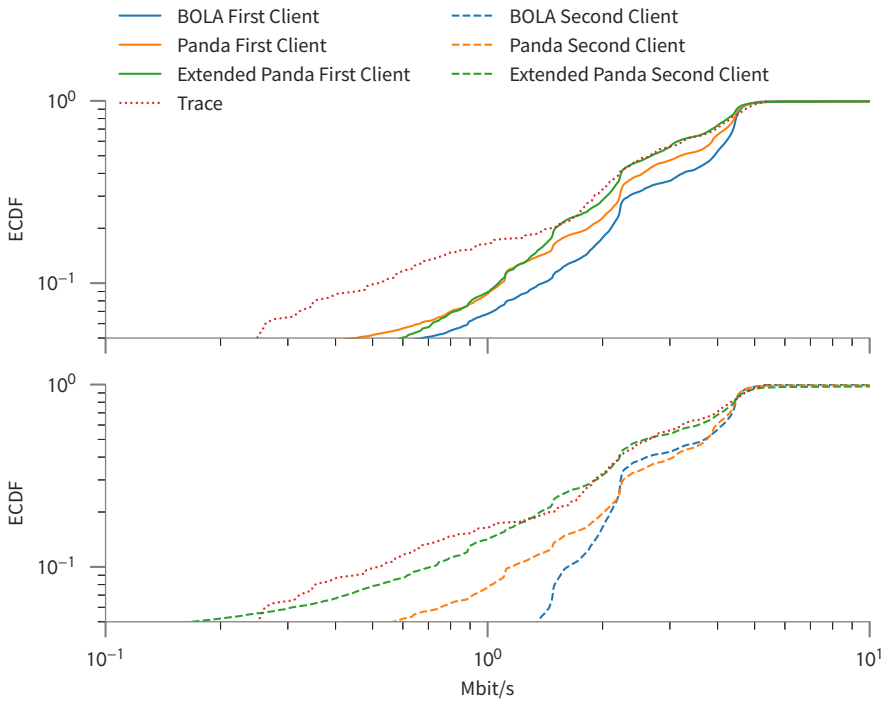


Figure 58: Comparison of the investigated AAs: BOLA, Panda, and extended Panda. For each, two clients play in sequential order applying the same bandwidth trace for each client. The cache size is limited to 1500 Segments. Server bandwidth was 4 Mbps, Caches bandwidth 10 Mbps, 20 Mbps, 30 Mbps respectively.

Table 16: Overall comparison of adaptation, video quality, and stalling metrics.

	BOLA		Panda		Panda (extended)	
Client	C_1	C_2	C_1	C_2	C_1	C_2
Adaptations [#]	19	18	80	79	20	26
Magnitude [level]	0.2	0.2	0.8	0.8	0.2	0.3
Stalling sum [sec]	2	4	7	13	0	0
Stalling [#]	2	4	5	6	0	0
Bitrate	2.3	2.1	2.0	2.2	2.3	2.5

When comparing these distributions between all clients and AAs, the underlying reasons for an improvement of the playback bitrate of the proposed extended Panda become evident; while all algorithms exhibit some degree of cache usage (segments may be also fetched from caches due to retransmissions in case of the first clients) we observe a comparatively large share of segments from bandwidths clustering around the network capacity of caches for the second client in case of the extended version of Panda. Therefore, overall a higher bandwidth for segments requests can be achieved beyond the bottleneck capacity of the server. Last, we also observe that roughly 1% of all throughput measurements are still overestimated,

due to burstiness in traffic patterns introduced by switches and the ICP congestion control (this is omitted in Figure 58).

6.6 *Practical Feasibility of NDN-based DASH Streaming*

Along with the previously presented evaluation of DASH Adaptation Algorithms performance in a virtualized testbed environment, in the following, we present a practical setup showing the discussed improvement in a tangible way.

In particular, this section highlights the advantages of seamless mobile handover, independent of the selected data source, as shown in a physical NDN testbed setup; here our developed system shows DASH streaming using a NDN-based transmission between data sources (Raspberry PI's) and mobile clients (Nexus 5 Android Phones) as video players.

With this system, the goal is to investigate the practical implication of underlying networking concepts of NDN and their potential impact on DASH streaming applications. Here, we focus on the particular use cases where benefits of NDN are most likely: First, by changing the location of the mobile clients, thus registering to different access points, routes to the Raspberry Pi data hosts will change their costs (e. g., in terms of round trip time (RTT))during runtime (see Figure 59). In a regular streaming scenario, the requests for the DASH playback would continue unaltered and the routes to data hosts with lower costs are neglected. In NDN, depending on the forwarding strategy used, the *best* route is dynamically selected for each request, providing a better Quality of Service (QoS). Next, the system

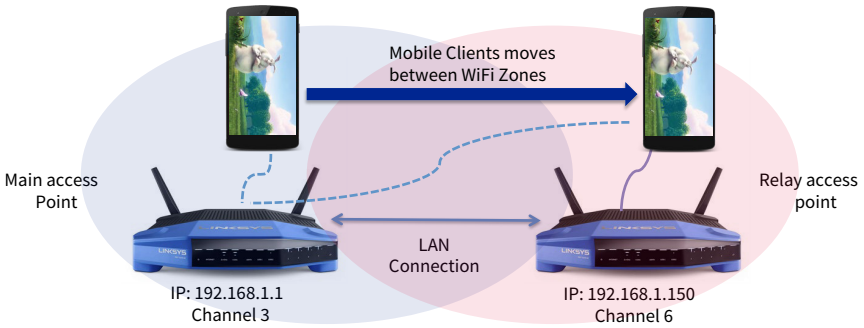


Figure 59: Process of a client-based wireless handover in NDN

also shows NDNs resilience when inducing node failures. Here, a regular DASH streaming system would not provide continuous playback without reinitializing the MPD providing an updated data host IP address. In contrast to this, the playback session in the demonstrated scenario provides a continuous playback as the identification of hosts is handled by the Named Data Networking Forwarding Daemon which abstracts the knowledge of hosts from the client.

As part of this demonstration, a web-based User Interface shows live information regarding the currently used routes, based on the selectable forwarding strategy, as well as node utilization in real time. The user can configure different routing strategies, e. g., `best-route`, `broadcast`, or `ncc`. In the first case, the Interest is forwarded to the lowest-cost next hop, which reduces the overall load on the system.

In the second case, the interest is forwarded to all eligible next hops, allowing for a higher resilience in the case of node failure or handovers. The last case shows the request behavior similar to the evaluations presented previously. Implications of the changes are evident on the overall system utilization and performance as shown on the client devices. This practical tested implementation of DASH in NDN has been presented as part of a demonstration in subproject C3 of the Collaborative Research Centre "MAKI".[bo]

6.7 *Conclusions*

The last contribution, presented in this chapter of our dissertation, presents a systematic study of challenges for DASH in NDN; based on emulative evidence, generated in a first study using an extended topology to induce effects of distributed caches, we show that current Adaptation Algorithms such as BOLA and PANDA exhibit limited streaming performance in NDN, i. e. high stalling times.

We identify the cause to be variations in the traffic measurements—underlying all standard DASH Adaptation Algorithms—which are more pronounced in NDNs than in traditional network setups: in an NDN each request can be served by any suitable NDN repository, DASH clients are not aware of it. Thus, the resulting variation in bandwidth from a client's perspective leads to sub-optimal adaptations decisions. Hence, new approaches for DASH adaptation algorithms considering the specific NDN context are necessary.

Given this initial analysis, we propose such a new concept for more fine-granular and precise measurements based on NDN chunks instead of DASH segments, as well as the use of additional information about the chunk sizes. By doing so, we achieve an improvement in the QoE over the existing AA PANDA. In general, the proposed extension can also be used for other throughput-based AAs in NDNs.

While the contributions in this work are demonstrated for the example of NDNs, the extensions to the PANDA AA regarding a reduction of adaptations steps and improved segment size awareness are general, i. e., they may also be deployed in the context of TCP-based DASH.

We found that the observed performance is strongly related to specific combinations of configurations, including the cache size and the used congestion control algorithm. We conclude that such mechanisms in each network layer have to be considered in DASH configurations. This is part of future studies.

In addition to this emulative study, in a practical testbed for DASH in NDN, we have presented a first application that directly shows the benefits of NDN for DASH-based mobile video streaming, in particular during handover between discrete networks. Here, playback can, in contrast to TCP/IP-based DASH, continue uninterrupted given the content-centric request model of NDN.

With regard to the domain of future network concepts, it is important to note that a symbiotic relation of the presented NDN-based video streaming concepts with regard to other upcoming concepts is plausible. NDNs multicast forwarding strategies, that, when implemented naïvely, do not fully utilize potentials for mobile users as transmission can still lead to congestion of the wireless medium

[bo] https://www.maki.tu-darmstadt.de/sfb_maki/ueber_maki/index.de.jsp

due to unicast-style connections on the PHY-Layer. Here, similar to the conceptual practical presentation of DASH in NDN, a demonstration of flexible PHY-Layer support for Scalable Video Coding (SVC)-based live streaming, i. e., using software-defined multicast mechanisms on the `PHY-layer` on mobile devices was presented in [Sto+15b].[22] By integrating these concepts with, e. g., our previously published work on Software-Defined Radio (SDR)-based control of PHY-Layer properties, such potentials can be realized more using a flexible multicast approach that can be integrated with the given forwarding strategies of NDN for live video upload and download from mobile devices.

[22] *for a detailed discussion of this concept and the respective demonstration, we refer to the dissertation by [Sch18].*

7 Conclusions and Outlook

This dissertation presented research towards improving users' Quality of Experience (QoE) in current and future video applications on the Internet. In particular, we presented and discussed our results in three central areas for such applications: Mobile Video Composition (MVC), Over-The-Top (OTT) Dynamic Adaptive Streaming over HTTP (DASH), and applying these concepts in future Named Data Networking (NDN) networks. To summarize our findings, we highlight key results of this work, our concrete contributions made, and close with remarks on future research directions.

7.1 Thesis Summary

Along the lines of the central subject of research in this thesis, i. e., the systematic study of video-centric applications towards attaining measurable—in form of QoE indicators—and realistically viable performance improvements for upload and distribution of video streams, applicable in current and future Internet architectures, we now summarize our results for the three presented areas of investigation.

As the first contribution of this dissertation in Chapter 4, we presented an analysis of user behavior and video quality in the live Mobile Video Broadcasting Service (MBS) YouNow. The results obtained in this study indicate a low overall quality in live mobile video upload that depends on the given connection type. Here, mobile networks were the main indicator for a low stream quality. To address such shortcoming, we presented a concept for the automatic creation of video compositions using such live User-Generated Videos (UGVs) that addresses such issues as follows: first, based on a context-based evaluation of streams *on the devices*, we can infer a measure of quality at the source and select relevant streams so that a composition with an overall higher QoE can be achieved. Secondly, the proposed approach was examined in a field study, where five sets of live uploads were produced with multiple users and devices across heterogeneous mobile networks. Finally, we verified the improved performance of the generated streams using a crowd-sourced user study. As a result, we have presented the first approach for low overhead generation of MVC from heterogeneous mobile sources that minimizes resource utilization on networks and devices.

In Chapter 5 on cross-layer evaluation of DASH mechanisms, we continued with the investigation of user experience in Video Streaming Systems (VSSs), thus focusing on the distribution of video content in the current Internet architecture. Here we argue that DASH VSSs performance strongly depends on the cross-layer configuration space instead of single system aspects such as Adaptation Algorithms (AAs). To verify our research hypothesis, we first introduced a concept for generat-

ing an extensive set of evaluations with regard to the cross-layer mechanisms used in DASH VSS, and to reproducibly evaluate the performance of real-world DASH systems. Using this approach, we derive a broad set of experiments covering all relevant aspects of cross-layer DASH VSS configuration parameters, that allow to understand performance aspects related to, e. g., the Transmission Control Protocol (TCP) Congestion Control (CC), Adaptation Algorithms, and DASH Players within heterogeneous network environments. Our derived results not only verify our initial hypothesis on the need to consider the full set of cross-layer configuration space in DASH experiments but also show that certain concrete configurations provide improved performance in terms of QoE. Our findings are extended by a study on how to adapt DASH systems given diverse and changing network environments by the reconfiguration of DASH mechanisms using learned transition boundaries between such configurations.

The last contribution presented Chapter 6 extends the conducted research on DASH towards the applicability for the future Internet architecture denoted as NDN. Using a similar systemic approach, we begin with an emulative study of the impact of caching, as a central aspect of NDN, on the performance of DASH Adaptation Algorithms. Here, in contrast to previous work, we use an extended topology size comprising several caches and clients towards identifying the effects of distributed caches on adaptation choices. Our experiments indicate that current Adaptation Algorithms such as BOLA and PANDA exhibit limited streaming performance, i. e., high stalling times due to the loss of end-to-end connection-oriented design compared to traditional DASH VSSs.

We use the derived insights to propose a new concept for AAs tailored to the requirements of NDN networks. By using fine-granular and precise measurements based on a probe-gap model, our suggested extension for DASH AAs achieves improved performance by reducing stalling and increasing streaming bitrates as compared to current DASH AAs. This concept of using NDN for DASH is also presented in a practical demonstration that presents improved support for mobile handovers with mobile clients.

7.2 Contributions

At the beginning of this work, we identified three synergetic research areas to be addressed in this dissertation—selected to address relevant limitations in VSSs for current and future use cases. First, addressing the full spectrum of user-centric video distribution, from live uploads to on-demand streaming, and second, to identify concrete possibilities for performance improvement in the current and future Internet. In the following, we will address how the before-mentioned contributions correspond and integrate to closing the identified research gaps.

Our first goal was to tackle the insufficient efficiency in dynamic creation and distribution of user-generated video, in particular for MVC. Here, the context-based evaluation of streams, using sensor, network, and activity measurements, allowed us to increase efficiency by mitigating the need to upload all generated streams to be considered in compositions. Instead, our approach uses a quality indicator based on sensor readings of the source device only (i. e., forgoing content

analysis) and was shown to improve the quality of compositions over the baseline. This enables to drastically reduce the overhead for data transmission and battery consumption on mobile devices as compared to uploading and analyzing all candidate video streams.

Secondly, this work was conducted to address achieving a high user satisfaction for over-the-top video distribution using DASH, which is of high relevance for streaming Video on Demand (VoD) but also MBS content. To this end, we motivated the problem of interdependence between a network's characteristics and configurations and developed an approach to understand these relations by conducting large-scale emulations. By identifying Pareto-optimal DASH configurations as well as proposing how to select configurations depending on the network context we provide evidence on how to realize a consistent high QoE with DASH in heterogeneous network conditions.

The last research goal then was to explore future networking concepts for adaptive video streaming applications. Here, our focus was to use NDN as a promising concept for future networks that tackles many of the current shortcomings in the existing Internet architecture. Along with our previous focus on DASH for the distribution of content, we identified crucial aspects for the design of video-centric applications when using the NDN architecture. We finally provide a new method for DASH Adaptation Algorithms allowing a robust performance in such network architectures and conclude our investigation of this domain by giving a concrete example for benefits in a use case for mobile network handovers in DASH over NDNs.

7.3 Outlook

In the following, we will discuss future areas of investigation in conjunction with the work presented in this dissertation.

7.3.1 Extending DASH Research using other Players

To extend the findings of our work on extensive emulations for DASH, including further players, i. e. commercial software such as Bitmovin's HTML5 Player would further highlight, how design choices in these players influence streaming performance. In particular, varying methods for throughput estimation, buffer management, and request handling (such as overlapping requests) may be of particular interest for the evolvement of adaptive streaming software.

7.3.2 Applying DASH Reconfigurations in Practice

As one of the outcomes of our work, we have identified the potential of reconfiguration in DASH VSSs to improve QoE for a given set of environment parameters, such as available bandwidth (and variations thereof), delay, and loss. The next step is to practically validate such transitions by first efficiently identifying such environment changes (e. g., [Ric+18]) and executing given transitions towards attaining concrete improvements in streaming performance. While in this work, we have relied on simple models to learn such relations, it is possibly beneficial to include

more environmental dimensions and better performing learning algorithms to identify configuration environments for DASH.

7.3.3 NDNs for Mobile Video Composition

With this work, we have shown two aspects that enable future mobile video composition applications. First, identifying and evaluating relevant content on the uploading source, and secondly, showing the feasibility of DASH in NDN. By combining these concepts in future work, users can profit from creating individual compositions by selecting and streaming content produced on mobile devices dynamically. In other words, as in NDN, users do not need to directly identify the hosts for requesting content, merely being aware of content named by, i. e. event, location, user or topic, a composed video stream may then only consist of a selection of segments with a given name. This concept would profit from the implicit caching in NDN so that, if a certain view becomes widely requested, content is automatically cached on the given requests paths.

7.3.4 Merging Context and Content in Mobile Video Composition

At this point, our proposed concept for Mobile Video Composition (MVC) relies solely on sensor-based measurements for the given recording context on the devices for evaluating the candidate streams' relevance and quality. We acknowledge, however, that this approach does not identify cases in which the content itself needs to be investigated to understand detrimental aspects on the recording quality, e. g., harmful occlusions, as discussed in [Wil16]. In a future system, integrating both concepts, i. e., context and content analysis, and thus balancing overhead and accuracy for stream source evaluations promises to provide superior results for MVC.

Bibliography

[Ahl+12] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Börje Ohlman. "A Survey of Information-Centric Networking." In: *IEEE Communications Magazine* 50.7 (07/2012). 01203, pp. 26–36 (cit. on p. 88).

[AN11] Shane Alcock and Richard Nelson. "Application Flow Control in YouTube Video Streams." In: *ACM SIGCOMM Computer Communication Review* 41.2 (2011), pp. 24–30 (cit. on pp. 25 sq.).

[Año+18] Javier Añorga, Saioa Arrizabalaga, Beatriz Sedano, Jon Goya, Maykel Alonso-Arce, and Jaizki Mendizabal. "Analysis of YouTube's Traffic Adaptation to Dynamic Environments." In: *Multimedia Tools and Applications* 77.7 (04/01/2018), pp. 7977–8000 (cit. on p. 30).

[Awi+13] Suphakit Awiphan, Takeshi Muto, Yu Wang, Zhou Su, and Jiro Katto. "Video Streaming over Content Centric Networking: Experimental Studies on PlanetLab." In: *Proceesings of Computing, Communications and IT Applications Conference (ComComAp)*. IEEE, 2013, pp. 19–24 (cit. on pp. 34, 90).

[Bal+12] Athula Balachandran, Vyas Sekar, Aditya Akella, Srinivasan Seshan, Ion Stoica, and Hui Zhang. "A Quest for an Internet Video Quality-of-Experience Metric." In: *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. HotNets-XI. 2012, pp. 97–102 (cit. on p. 20).

[Bao+13] Xuan Bao, Songchun Fan, Alexander Varshavsky, Kevin Li, and Romit Roy Choudhury. "Your Reactions Suggest You Liked the Movie: Automatic Content Rating via Reaction Sensing." In: *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2013, pp. 197–206 (cit. on p. 28).

[Bha+15] Divyashri Bhat, Cong Wang, Amr Rizk, and Michael Zink. "A Load Balancing Approach for Adaptive Bitrate Streaming in Information Centric Networks." In: *2015 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*. 06/2015, pp. 1–6 (cit. on pp. 36, 90).

[BR10] Xuan Bao and Romit Roy Choudhury. "Movi: Mobile Phone Based Video Highlights via Collaborative Sensing." In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. ACM, 2010, pp. 357–370 (cit. on p. 28).

[Bru+13] Kjell Brunnström, Sergio Ariel Beker, Katrien De Moor, Ann Dooms, et al. *Qualinet White Paper on Definitions of Quality of Experience*. 2013 (cit. on p. 22).

[BRZ17] Divyashri Bhat, Amr Rizk, and Michael Zink. "Not so QUIC: A Performance Study of DASH over QUIC." In: *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2017, pp. 13–18 (cit. on p. 36).

[Cal+14] Christian Callegari, Stefano Giordano, Michele Pagano, and Teresa Pepe. "A Survey of Congestion Control Mechanisms in Linux TCP." In: *Distributed Computer and Communication Networks*. Communications in Computer and Information Science. Springer, Cham, 2014, pp. 28–42 (cit. on p. 7).

[Car+16] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. "BBR: Congestion-Based Congestion Control." In: *ACM Queue* 14, September-October (2016), pp. 20–53 (cit. on p. 7).

[CGM12] G. Carofiglio, M. Gallo, and L. Muscariello. "ICP: Design and Evaluation of an Interest Control Protocol for Content-Centric Networking." In: *Proceedings of the 2012 IEEE INFOCOM Workshop*. 03/2012, pp. 304–309 (cit. on p. 91).

[Cha+07] Meeyoung Cha, Haewoon Kwak, Pablo Rodriguez, Yong-Yeol Ahn, and Sue Moon. "I Tube, You Tube, Everybody Tubes: Analyzing the World's Largest User Generated Content Video System." In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. ACM, 2007, pp. 1–14 (cit. on p. 38).

[Che+16] Jiachen Chen, Mayutan Arumaithurai, Xiaoming Fu, and K. K. Ramakrishnan. "SAID: A Control Protocol for Scalable and Adaptive Information Dissemination in ICN." In: *Proceedings of the 3rd ACM Conference on Information-Centric Networking*. New York, NY, USA: ACM, 2016, pp. 11–20 (cit. on p. 91).

[Cis17] Cisco. *The Zettabyte Era: Trends and Analysis*. 06/2017. URL: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/vni-hyperconnectivity-wp.pdf (cit. on p. 2).

[CPW11] G. Cermak, M. Pinson, and S. Wolf. "The Relationship Among Video Quality, Screen Resolution, and Bit Rate." In: *IEEE Transactions on Broadcasting* 57.2 (06/2011), pp. 258–262 (cit. on p. 12).

[Cra+98] Eric Crawley, H Sandick, R Nair, and B Rajagopalan. "A Framework for QoS-Based Routing in the Internet." In: *RFC 2386* (1998) (cit. on p. 19).

[Cri+12] Francesco Cricri, IgorD.D. Curcio, Sujeet Mate, Kostadin Dabov, and Moncef Gabbouj. "Sensor-Based Analysis of User Generated Video for Multi-Camera Video Remixing." In: ed. by Klaus Schoeffmann, Bernard Merialdo, AlexanderG. Hauptmann, Chong-Wah Ngo, Yiannis Andreopoulos, and Christian Breiteneder. Berlin,

Heidelberg: Springer Berlin Heidelberg, 2012, pp. 255–265 (cit. on pp. 28 sq.).

[DAS17] DASH-IF. *DASH Industry Forum: DASH-AVC/264 Test Cases and Vectors*. 09/2017. URL: http://dashif.org/wp-content/uploads/2015/04/DASH-AVC-264-Test-Vectors-v09-CommunityReview.pdf (cit. on pp. 90, 96).

[De +14] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. "TAPAS: A Tool for rApid Prototyping of Adaptive Streaming Algorithms." In: *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*. New York, NY, USA: ACM, 2014, pp. 1–6 (cit. on p. 33).

[Dij82] Edsger W. Dijkstra. "On the Role of Scientific Thought." In: *Selected Writings on Computing: A Personal Perspective*. Texts and Monographs in Computer Science. Springer, New York, NY, 1982, pp. 60–66 (cit. on p. 6).

[Dob+13] Florin Dobrian, Asad Awan, Dilip Joseph, Aditya Ganjam, et al. "Understanding the Impact of Video Quality on User Engagement." In: *Communications of the ACM* 56.3 (2013), pp. 91–99 (cit. on p. 72).

[Dua+17] Z. Duanmu, K. Zeng, K. Ma, A. Rehman, and Z. Wang. "A Quality-of-Experience Index for Streaming Video." In: *IEEE Journal of Selected Topics in Signal Processing* 11.1 (02/2017), pp. 154–166 (cit. on p. 85).

[EEJ08] Arvid Engstrom, Mattias Esbjornsson, and Oskar Juhlin. "Mobile Collaborative Live Video Mixing." In: *Proceedings of the ACM International Conference on Human Computer Interaction with Mobile Devices and Services*. ACM, 2008, pp. 157–166 (cit. on p. 27).

[Eit+13] Philipp M. Eittenberger, Michael Hamatschek, Marcel Großmann, and Udo R. Krieger. "Monitoring Mobile Video Delivery to Android Devices." In: *Proceedings of the 4th ACM Multimedia Systems Conference*. MMSys '13. New York, NY, USA: ACM, 2013, pp. 119–124 (cit. on p. 30).

[El +15] Ali El Essaili, Zibin Wang, Eckehard Steinbach, and Liang Zhou. "QoE-Based Cross-Layer Optimization for Uplink Video Transmission." In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 12.1 (2015), p. 2 (cit. on p. 27).

[Eve+14] Kristian Evensen, Tomas Kupka, Haakon Riiser, Pengpeng Ni, et al. "Adaptive Media Streaming to Mobile Devices: Challenges, Enhancements, and Recommendations." In: *Advances in Multimedia 2014* (2014), pp. 1–21 (cit. on p. 32).

[Fin+11] Alessandro Finamore, Marco Mellia, Maurizio M Munafò, Ruben Torres, and Sanjay G Rao. "Youtube Everywhere: Impact of Device and Infrastructure Synergies on User Experience." In: *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement*. ACM, 2011, pp. 345–360 (cit. on p. 26).

[Frö+17] Alexander Frömmgen, Amr Rizk, Tobias Erbshäußer, Max Weller, et al. "A Programming Model for Application-Defined Multipath TCP Scheduling." In: *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference*. New York, NY, USA: ACM, 2017, pp. 134–146 (cit. on pp. 36, 63, 83).

[Frö+18] Alexander Frömmgen, Denny Stohr, Boris Koldehofe, and Amr Rizk. "Don't Repeat Yourself: Seamless Execution and Analysis of Extensive Network Experiments." In: 2018. arXiv: 1802.03455 (cit. on pp. 62 sq.).

[Fu+13] Bo Fu, Dirk Staehle, Gerald Kunzmann, Eckehard Steinbach, and Wolfgang Kellerer. "QoE-Aware Priority Marking and Traffic Management for H.264/SVC-Based Mobile Video Delivery." In: *Proceedings of the 8th ACM Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks*. PM2HW2N '13. New York, NY, USA: ACM, 2013, pp. 173–180 (cit. on p. 35).

[FZX16] Bohao Feng, Huachun Zhou, and Qi Xu. "Mobility Support in Named Data Networking: A Survey." In: *EURASIP Journal on Wireless Communications and Networking* 2016 (09/15/2016), p. 220 (cit. on p. 88).

[Gab91] Richard P. Gabriel. *Worse Is Better*. 1991. URL: https://www.dreamsongs.com/WIB.html (Last accessed on March 8, 06/03/2018) (cit. on p. 13).

[GJS03] Janusz Gozdecki, Andrzej Jajszczyk, and Rafal Stankiewicz. "Quality of Service Terminology in IP Networks." In: *IEEE Communications Magazine* 41.3 (03/2003), pp. 153–159 (cit. on p. 20).

[GPB18] D. Ghadiyaram, J. Pan, and A. C. Bovik. "Learning a Continuous-Time Streaming Video QoE Model." In: *IEEE Transactions on Image Processing* 27.5 (05/2018), pp. 2257–2271 (cit. on p. 85).

[GSW13] Reinhard Grandl, Kai Su, and Cedric Westphal. "On the Interaction of Adaptive Video Streaming with Content-Centric Networking." In: *Proceedings of the 20th International Packet Video Workshop* (2013) (cit. on pp. 34, 90).

[Han+12] Nikhil Handigol, Heller Brandon, Vimalkumar Jeyakumar, Bob Lantz, and Nick McKeown. "Reproducible Network Experiments Using Container-Based Emulation." In: *Proceedings of the International Conference on Emerging Networking Experiments and Technologies*. ACM, 2012, pp. 253–264 (cit. on p. 63).

[Han+16] Bo Han, Feng Qian, Lusheng Ji, and Vijay Gopalakrishnan. "MP-DASH: Adaptive Video Streaming Over Preference-Aware Multipath." In: *Proceedings of the 12th International on Conference on Emerging Networking EXperiments and Technologies*. CoNEXT '16. New York, NY, USA: ACM, 2016, pp. 129–143 (cit. on p. 36).

[Hoß+11] Tobias Hoßfeld, Sebastian Biedermann, Raimund Schatz, Alexander Platzer, Sebastian Egger, and Markus Fiedler. "The Memory Effect and Its Implications on Web QoE Modeling." In: Proceedings of the 23rd International Teletraffic Congress (ITC). 2011, pp. 103–110 (cit. on p. 23).

[Hoß+13] Tobias Hoßfeld, Raimund Schatz, Ernst Biersack, and Louis Plissonneau. *Internet Video Delivery in YouTube: From Traffic Measurements to Quality of Experience*. Vol. 7754. Data Traffic Monitoring and Analysis: From Measurement, Classification, and Anomaly Detection to Quality of Experience. Springer, 2013. 264 pp. (cit. on pp. 24, 68).

[Hoß+14a] Tobias Hoßfeld, Christian Keimel, Matthias Hirth, Bruno Gardlo, et al. "Best Practices for QoE Crowdtesting: QoE Assessment with Crowdsourcing." In: *IEEE Transactions on Multimedia* 16.2 (2014), pp. 541–558 (cit. on p. 23).

[Hoß+14b] Tobias Hoßfeld, Michael Seufert, Christian Sieber, and Thomas Zinner. "Assessing Effect Sizes of Influence Factors Towards a QoE Model for HTTP Adaptive Streaming." In: Proceedings of the Sixth International Workshop on Quality of Multimedia Experience (QoMEX). 2014, pp. 111–116 (cit. on pp. 24, 68, 79).

[Hoß+17] Tobias Hoßfeld, Poul E. Heegaard, Lea Skorin-Kapov, and Martin Varela. "No Silver Bullet: QoE Metrics, QoE Fairness, and User Diversity in the Context of QoE Management." In: *Proceeings of the Ninth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2017, pp. 1–6 (cit. on p. 85).

[HS02] Ningning Hu and Peter Steenkiste. *Estimating Available Bandwidth Using Packet Pair Probing*. 2002. URL: http://www.dtic.mil/docs/citations/ADA461170 (cit. on p. 50).

[Hua+12] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. "Confused, Timid, and Unstable: Picking a Video Streaming Rate Is Hard." In: *Proceedings of the 2012 ACM Conference on Internet Measurement*. ACM, 2012, pp. 225–238 (cit. on pp. 31 sq.).

[Hua+14] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. "A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service." In: *Proceedings of the ACM Conference of the Special Interest Group on Data Communication*. SIGCOMM '14. ACM, 2014, pp. 187–198 (cit. on pp. 30 sqq., 36).

[Iba+09] Eva Ibarrola, Fidel Liberal, Ianire Taboada, and Rodrigo Ortega. "Web QoE Evaluation in Multi-Agent Networks: Validation of ITU-T G.1030." In: *2009 Fifth International Conference on Autonomic and Autonomous Systems*. IEEE, 2009, pp. 289–294 (cit. on p. 23).

[Ish+15] Yuya Ishizu, Kenji Kanai, Jiro Katto, Hidenori Nakazato, and Marie Hirose. "Energy-Efficient Video Streaming over Named Data Networking Using Interest Aggregation and Playout Buffer Control." In: *Proceedings of the 2015 IEEE International Conference on Data Science and Data Intensive Systems (DSDIS)*. 2015, pp. 318–324 (cit. on p. 35).

[ITU08] ITU-T. *P.910: Subjective Video Quality Assessment Methods*. 2008, p. 42 (cit. on p. 23).

[ITU12] ITU-T. *BT.500: Methodology for the Subjective Assessment of the Quality of Television Pictures*. 2012 (cit. on p. 23).

[Jac+09] Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. "Networking Named Content." In: *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. CoNEXT '09. Rome, Italy: ACM, 2009, pp. 1–12 (cit. on p. 10).

[Jac88] Van Jacobson. "Congestion Avoidance and Control." In: *Proceeding of the SIGCOMM '88 Symposium Proceedings on Communications Architectures and Protocols*. Vol. 18. ACM, 1988, pp. 314–329 (cit. on p. 50).

[JE08] Arulsaravana Jeyaraj and Magda El Zarki. "A Real-Time Cross-Layer Design of the Multimedia Application Layer with a Mimo Based Wireless Physical Layer." In: *Proceedings of the 3rd International Symposium on Wireless Pervasive Computing*. 2008, pp. 455–458 (cit. on p. 35).

[Joh+09] Dag Johansen, Havard Johansen, Tjalve Aarflot, Joseph Hurley, et al. "DAVVI: A Prototype for the Next Generation Multimedia Entertainment Platform." In: Proceedings of the ACM International Conference on Multimedia. 2009, pp. 989–990 (cit. on p. 27).

[JR86] Raj Jain and Shawn A. Routhier. "Packet Trains–Measurements and a New Model for Computer Network Traffic." In: *IEEE Journal on Selected Areas in Communications* 4.6 (1986), pp. 986–995 (cit. on p. 50).

[JSC17] R. Jmal, G. Simon, and L. Chaari. "Network-Assisted Strategy for Dash over CCN." In: *Proceedings of the 2017 IEEE International Conference on Multimedia and Expo (ICME)*. 07/2017, pp. 13–18 (cit. on p. 36).

[JSZ14] Junchen Jiang, Vyas Sekar, and Hui Zhang. "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive." In: *IEEE/ACM Transactions on Networking* 22.1 (02/2014), pp. 326–340 (cit. on pp. 30, 32, 36, 95).

[JTM15] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi. "SARA: Segment Aware Rate Adaptation Algorithm for Dynamic Adaptive Streaming over HTTP." In: *Proceedings of the IEEE International Conference on Communication and Workshop*. 2015 (cit. on p. 95).

[JTM16] Parikshit Juluri, Venkatesh Tamarapalli, and Deep Medhi. "QoE Management in DASH Systems Using the Segment Aware Rate Adaptation Algorithm." In: *Proceedings of the 2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 04/2016, pp. 129–136 (cit. on pp. 67 sq.).

[KAB17] Jonathan Kua, Grenville Armitage, and Philip Branch. "A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP." In: *IEEE Communications Surveys & Tutorials* 19.3 (2017), pp. 1842–1866 (cit. on p. 29).

[Kan+14] L. Kang, P. Ye, Y. Li, and D. Doermann. "Convolutional Neural Networks for No-Reference Image Quality Assessment." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1733–1740 (cit. on p. 22).

[Kan+15] Kenji Kanai, Takeshi Muto, Jiro Katto, Shinya Yamamura, et al. "Performance Evaluation of Proactive Content Caching for Mobile Video through 50-User Field Experiment." In: *Proceedings of the IEEE Globecom Workshop*. 12/2015, pp. 1–6 (cit. on p. 35).

[Kes95] S. Keshav. "A Control-Theoretic Approach to Flow Control." In: *ACM SIGCOMM Computer Communication Review* 25.1 (01/1995). 01086, pp. 188–201 (cit. on p. 50).

[KF17] Sukhpreet Kaur Khangura and Markus Fidler. "Available Bandwidth Estimation from Passive TCP Measurements Using the Probe Gap Model." In: *Proceedings of IFIP Networking*. IEEE, 06/2017 (cit. on p. 93).

[Khu+18] Wasiur R. KhudaBukhsh, Bastian Alt, Sounak Kar, Amr Rizk, and Heinz Koeppl. "Collaborative Uploading in Heterogeneous Networks: Optimal and Adaptive Strategies." In: *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*, 2018 (cit. on p. 27).

[Kop+07] Teemu Koponen, Mohit Chawla, Byung-Gon Chun, Andrey Ermolinskiy, et al. "A Data-Oriented (and beyond) Network Architecture." In: *ACM SIGCOMM Computer Communication Review* 37.4 (2007), p. 181 (cit. on p. 9).

[Kre+16] Christian Kreuzberger, Benjamin Rainer, Herman Hellwagner, Laura Toni, and Pascal Frossard. "A Comparative Study of DASH Representation Sets Using Real User Characteristics." In: *Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. 2016, p. 4 (cit. on p. 68).

[Lan+17] Adam Langley, Janardhan Iyengar, Jeff Bailey, Jeremy Dorfman, et al. "The QUIC Transport Protocol." In: *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. 2017, pp. 183–196 (cit. on pp. 8, 36).

[LCK08] Mingzhe Li, Mark Claypool, and Robert Kinicki. "WBest: A Bandwidth Estimation Tool for IEEE 802.11 Wireless Networks." In: *Proceedings of the 33rd IEEE Conference on Local Computer Networks*. 2008, pp. 374–381 (cit. on p. 50).

[Li+14] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, et al. "Probe and Adapt: Rate Adaptation for HTTP Video Streaming at Scale." In: *IEEE Journal on Selected Areas in Communications* 32.4 (2014), pp. 719–733 (cit. on pp. 30, 89, 95).

[Li+16] Zhi Li, Anne Aaron, Ioannis Katsavounidis, Anush Moorthy, and Megha Manohara. *Toward A Practical Perceptual Video Quality Metric*. 2016. URL: http://techblog.netflix.com/2016/06/toward-practical-perceptual-video.html (Last accessed on March 8, 09/22/2016) (cit. on p. 21).

[Liu+13] Yaning Liu, Joost Geurts, Jean Charles Point, Stefan Lederer, et al. "Dynamic Adaptive Streaming over CCN: A Caching and Overhead Analysis." In: *IEEE International Conference on Communications* (June 2013 2013), pp. 3629–3633 (cit. on pp. 34, 88, 90, 92).

[LMT12] Stefan Lederer, Christopher Müller, and Christian Timmerer. "Dynamic Adaptive Streaming over HTTP Dataset." In: *Proceedings of the ACM Multimedia Systems Conference*. New York, NY, USA: ACM Press, 02/2012, p. 89 (cit. on p. 68).

[MC17] Sujeet Mate and Igor D.D. Curcio. "Automatic Video Remixing Systems." In: *IEEE Communications Magazine* 55.1 (01/2017), pp. 180–187 (cit. on pp. 28 sq.).

[MCC11] Ricky K. P. Mok, Edmond W. W. Chan, and Rocky K. C. Chang. "Measuring the Quality of Experience of HTTP Video Streaming." In: *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 05/2011, pp. 485–492 (cit. on p. 19).

[Mil+12] Konstantin Miller, Emanuele Quacchio, Gianluca Gennari, and Adam Wolisz. "Adaptation Algorithm for Adaptive Streaming over HTTP." In: *Proceedings of the ACM Packet Video Workshop* (2012), pp. 173–178 (cit. on p. 30).

[Mit92] J. Mitola. "Software Radios-Survey, Critical Evaluation and Future Directions." In: *Proceedings of NTC-92: National Telesystems Conference*. 00838. 05/1992, pp. 13/15–13/23 (cit. on p. 8).

[MKK15] Takeshi Muto, Kenji Kanai, and Jiro Katto. "Implementation Evaluation of Proactive Content Caching Using DASH-NDN-JS." In: *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 03/2015, pp. 2239–2244 (cit. on p. 35).

[MNA17] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. "Neural Adaptive Video Streaming with Pensieve." In: *Proceedings of ACM SIGCOMM '17*. New York, New York, USA: ACM Press, 2017, pp. 197–210 (cit. on pp. 30 sqq.).

[Mon+17] Abhijit Mondal, Satadal Sengupta, Bachu Rikith Reddy, M. J.V. Koundinya, et al. "Candid with YouTube." In: *Proceedings of the 27th NOSSDAV Workshop*. New York, New York, USA: ACM Press, 2017, pp. 19–24 (cit. on pp. 14 sq., 30 sq., 76, 83).

[MPF16] Stephen McQuistin, Colin Perkins, and Marwan Fayed. "TCP Hollywood: An Unordered, Time-Lined, TCP for Networked Multimedia Applications." In: *Proceedings of IFIP Networking*. IEEE, 05/2016, pp. 422–430 (cit. on p. 19).

[MR11] James Manyika and Charles Roxburgh. "The Great Transformer: The Impact of the Internet on Economic Growth and Prosperity." In: *McKinsey Global Institute* (2011) (cit. on p. 1).

[MVA16] Toni Maki, Martin Varela, and Doreid Ammar. "A Layered Model for Quality Estimation of HTTP Video from QoS Measurements." In: *Proceedings of SITIS 2015*. IEEE, 2016, pp. 591–598 (cit. on p. 32).

[Naz+14] Sajid Nazir, Ziaul Hossain, Raffaello Secchi, Matthew Broadbent, Andreas Petlund, and Gorry Fairhurst. "Performance Evaluation of Congestion Window Validation for DASH Transport." In: *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*. NOSSDAV '14. ACM, 2014, pp. 67–72 (cit. on p. 32).

[NHF07] A. Naghdinezhad, M. R. Hashemi, and O. Fatemi. "A Novel Adaptive Unequal Error Protection Method for Scalable Video over Wireless Networks." In: *Proceedings of the 2007 IEEE International Symposium on Consumer Electronics*. 06/2007, pp. 1–6 (cit. on p. 35).

[P1213] ITU-T P.1201. *Parametric Non-Intrusive Assessment of Audiovisual Media Streaming Quality. Amendment 2: New Appendix III – Use of ITU-T P.1201 for Non-Adaptive, Progressive Download Type Media Streaming*. 2013 (cit. on p. 24).

[Pos+14] Daniel Posch, Christian Kreuzberger, Benjamin Rainer, and Hermann Hellwagner. "Using In-Network Adaptation to Tackle Inefficiencies Caused by DASH in Information-Centric Networks." In: *Proceedings of the 2014 Workshop on Design, Quality and Deployment of Adaptive Video Streaming*. VideoNext '14. New York, NY, USA: ACM, 2014, pp. 25–30 (cit. on p. 36).

[Pos80] Jon Postel. *User Datagram Protocol*. STD 6. 04356 http://www.rfc-editor.org/rfc/rfc768.txt. RFC Editor, 08/1980 (cit. on p. 8).

[Pos81] Jon Postel. *Transmission Control Protocol*. STD 7. 10210 http://www.rfc-editor.org/rfc/rfc793.txt. RFC Editor, 09/1981 (cit. on p. 6).

[Pro+03] R. Prosad, C. Davrolis, M. Murray, and K.C. Claffy. "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools." In: *IEEE Network* 17.6 (11/2003), pp. 27–35 (cit. on p. 19).

[PS15] Karine Pires and Gwendal Simon. "YouTube Live and Twitch: A Tour of User-Generated Live Streaming Systems." In: *Proceedings of*

the ACM Multimedia Systems Conference. ACM, 03/2015, pp. 225–230 (cit. on pp. 26 sq.).

[PW04] M. H. Pinson and S. Wolf. "A New Standardized Method for Objectively Measuring Video Quality." In: *IEEE Transactions on Broadcasting* 50.3 (09/2004). 01137, pp. 312–322 (cit. on p. 30).

[Ram+14] Juan J Ramos-Muñoz, Jonathan Prados-Garzon, Pablo Ameigeiras, Jorge Navarro-Ortiz, and Juan M López-Soler. "Characteristics of Mobile Youtube Traffic." In: *IEEE Wireless Communications* 21.1 (2014), pp. 18–25 (cit. on p. 25).

[Ren+15] Yongmao Ren, Jun Li, Shanshan Shi, Lingling Li, and Xiangqing Chang. "An Interest Control Protocol for Named Data Networking Based on Explicit Feedback." In: *Proceedings of the Eleventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems*. ANCS '15. Washington, DC, USA: IEEE, 2015, pp. 199–200 (cit. on p. 91).

[RF16] Amr Rizk and Markus Fidler. "Queue-Aware Uplink Scheduling with Stochastic Guarantees." In: *Computer Communications* 84 (06/15/2016), pp. 63–72 (cit. on p. 27).

[Ric+16] Bjorn Richerzhagen, Julian Wulfheide, Heinz Koeppl, Andreas Mauthe, Klara Nahrstedt, and Ralf Steinmetz. "Enabling Crowdsourced Live Event Coverage with Adaptive Collaborative Upload Strategies." In: *Proceedings of the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, 06/2016, pp. 1–3 (cit. on p. 27).

[Ric+18] Nils Richerzhagen, Patrick Lieser, Bjorn Richerzhagen, Boris Koldehofe, Ioannis Stavrakakis, and Ralf Steinmetz. "Change as Chance: Transition-Enabled Monitoring for Dynamic Networks and Environments." In: *Proceedings of the Conference on Wireless On-Demand Network Systems and Services (WONS)*. IEEE, 02/2018, pp. 51–58 (cit. on p. 104).

[Ric11] Iain E Richardson. *The H. 264 Advanced Video Compression Standard*. John Wiley & Sons, 2011 (cit. on p. 11).

[Ric17] Björn Richerzhagen. "Mechanism Transitions in Publish/Subscribe Systems - Adaptive Event Brokering for Location-Based Mobile Social Applications." Ph.D. Thesis. Darmstadt: Technische Universität, 2017 (cit. on p. 83).

[Rii+13] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pal Halvorsen. "Commute Path Bandwidth Traces from 3G Networks." In: *Proceedinfs of the ACM Multimedia Systems Conference*. New York, New York, USA: ACM, 2013, pp. 114–118 (cit. on pp. 65 sq., 90).

[Rod+14] Demostenes Z Rodriguez, Wang Zhou, L Rosa Renata, and Bressan Graca. "The Impact of Video-Quality-Level Switching on User Quality of Experience in Dynamic Adaptive Streaming over HTTP."

In: *EURASIP Journal on Wireless Communications and Networking* (2014), p. 216 (cit. on p. 20).

[Rot+17] M. R. Rotinsulu, B. Susilo, A. Presekal, E. Pramono, and R. F. Sari. "Measuring Quality of Services (QoS) of Several Forwarding Strategies on Named Data Networking(NDN) Using ndnSIM." In: *Proceedings of the 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*. 11/2017, pp. 45–49 (cit. on p. 36).

[RPH16] B. Rainer, D. Posch, and H. Hellwagner. "Investigating the Performance of Pull-Based Dynamic Adaptive Streaming in NDN." In: *IEEE Journal on Selected Areas in Communications* 34.8 (08/2016), pp. 2130–2140 (cit. on pp. 35 sq.).

[Rüc16] Julius Rückert. "Large-Scale Live Video Streaming over the Internet : Efficient an Flexible Content Delivery Using Network and Application-Layer Mechanisms." Ph.D. Thesis. München: Dr. Hut, 2016 (cit. on p. 87).

[RWS17] Abinesh Ramakrishnan, Cedric Westphal, and Jonnahtan Saltarin. "Adaptive Video Streaming over CCN with Network Coding for Seamless Mobility." In: *Proceedings of ISM 2016*. IEEE, 12/2017, pp. 238–242 (cit. on p. 88).

[Sai+12] Mukesh Kumar Saini, Raghudeep Gadde, Shuicheng Yan, and Wei Tsang Ooi. "MoViMash: Online Mobile Video Mashup." In: *Proceedings of the 20th ACM International Conference on Multimedia*. MM '12. Nara, Japan: ACM Press, 10/2012, pp. 139–148 (cit. on p. 28).

[Sai+13] Mukesh K. Saini, Seshadri Venkatagiri, Wei T. Ooi, and Mun C. Chan. "The Jiku Mobile Video Dataset." In: *Proceedings of the ACM Multimedia Systems Conference*. 2013 (cit. on p. 28).

[Sam+17] Jacques Samain, Giovanna Carofiglio, Luca Muscariello, Michele Papalini, et al. "Dynamic Adaptive Video Streaming: Towards a Systematic Comparison of ICN and TCP/IP." In: *IEEE Transactions on Multimedia* 19.10 (10/2017), pp. 2166–2181 (cit. on pp. 33 sq., 36, 90 sq.).

[Sch+96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. RFC 1889. 10045. RFC Editor, 01/1996 (cit. on p. 14).

[Sch18] Matthias Schulz. "Teaching Your Wireless Card New Tricks: Smartphone Performance and Security Enhancements Through Wi-Fi Firmware Modifications." Darmstadt: Technische Universität, 2018 (cit. on p. 101).

[SCP13] L. Saino, C. Cocora, and G. Pavlou. "CCTCP: A Scalable Receiver-Driven Congestion Control Protocol for Content Centric Networking." In: *2013 IEEE International Conference on Communications (ICC)*. 06/2013, pp. 3775–3780 (cit. on pp. 90 sq.).

[SCZ12] Beomjoo Seo, Weiwei Cui, and Roger Zimmermann. "An Experimental Study of Video Uploading from Mobile Devices with HTTP Streaming." In: *Proceedings of ACM MMSys*. New York, New York, USA: ACM Press, 02/2012, p. 215 (cit. on p. 27).

[Seu+15] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hossfeld, and Phuoc Tran-gia. "A Survey on Quality of Experience of HTTP Adaptive Streaming." In: *IEEE Communications Surveys & Tutorials* 17.1 (2015), pp. 469–492 (cit. on pp. 20, 78).

[Seu+17] Michael Seufert, Nikolas Wehner, Florian Wamser, Pedro Casas, Alessandro D'Alconzo, and Phuoc Tran-Gia. "Unsupervised QoE Field Study for Mobile YouTube Video Streaming with YoMoApp." In: *Proceedings of QoMEX*. IEEE, 05/2017, pp. 1–6 (cit. on p. 26).

[Shr+10] Prarthana Shrestha, Peter de With, Hans Weda, Mauro Barbieri, and Emile Aarts. "Automatic Mashup Generation from Multiple-Camera Concert Recordings." In: *Proceedings of the ACM International Conference on Multimedia*. Firenze, Italy: ACM, 2010, pp. 541–550 (cit. on pp. 28, 53).

[SLC07] David Soldani, Man Li, and Renaud Cuny. *QoS and QoE Management in UMTS Cellular Systems*. John Wiley & Sons, 2007 (cit. on p. 23).

[SMK16] Matti Siekkinen, Enrico Masala, and Teemu Kamarainen. "Anatomy of a Mobile Live Streaming Service: The Case of Periscope." In: *Proceedings of the ACM Internet Measurement Conference*. 2016 (cit. on p. 27).

[SMN17] Matti Siekkinen, Enrico Masala, and Jukka Nurminen. "Optimized Upload Strategies for Live Scalable Video Transmission from Mobile Devices." In: *IEEE Transactions on Mobile Computing* (2017), p. 14 (cit. on p. 27).

[SN95] Ralf Steinmetz and Klara Nahrstedt. *Multimedia: Computing, Communications, and Applications*. Prentice-Hall, 1995 (cit. on p. 11).

[Spi18] Kevin Spiteri. "From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player." In: *Proceedings of ACM MMSys*. 2018 (cit. on p. 83).

[SRL98] Henning Schulzrinne, Anup Rao, and Robert Lanphier. *Real Time Streaming Protocol (RTSP)*. RFC 2326. RFC Editor, 04/1998 (cit. on p. 14).

[Sta11] Tania Stathaki. *Image Fusion: Algorithms and Applications*. Elsevier, 2011. 520 pp. (cit. on p. 21).

[Sto+15a] Denny Stohr, Tao Li, Stefan Wilk, Silvia Santini, and Wolfgang Effelsberg. "An Analysis of the YouNow Live Streaming Platform." In: *Proceedings of the 9th IEEE Workshop on Network Measurements (WNM)*. B1, C03. Clearwater, USA, 10/2015 (cit. on p. 37).

[Sto+15b] Denny Stohr, Matthias Schulz, Matthias Hollick, and Wolfgang Effelsberg. "APP and PHY in Harmony: Demonstrating Scalable Video Streaming Supported by Flexible Physical Layer Control." In: *Proceedings of IEEE WoWMoM 2015 (Demos)*. The first two authors contributed equally. Boston, USA: IEEE, 06/2015, pp. 1–3 (cit. on p. 101).

[Sto+16a] Denny Stohr, Fares Beji, Rahul Dwarakanath, Ralf Steinmetz, and Wolfgang Effelsberg. "Mobile NDN-Based Dynamic Adaptive Streaming over HTTP." In: *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*. (Demo) C03. Duabi: IEEE, 2016, p. 3 (cit. on pp. 87 sq.).

[Sto+16b] Denny Stohr, Alexander Frömmgen, Jan Fornoff, Michael Zink, Alejandro Buchmann, and Wolfgang Effelsberg. "QoE Analysis of DASH Cross-Layer Dependencies by Extensive Network Emulation." In: *Proceedings of the SIGCOMM Internet-QoE Workshop*. Florianopolis: ACM, 2016 (cit. on pp. 62, 69).

[Sto+16c] Denny Stohr, Stefan Wilk, Iva Toteva, Wolfgang Effelsberg, and Ralf Steinmetz. "Context Not Content: A Novel Approach to Real-Time User-Generated Video Composition." In: *Proceedings of the 2016 IEEE International Symposium on Multimedia*. San Jose, CA, USA: IEEE, 2016, p. 6 (cit. on pp. 28, 37).

[Sto+17a] Denny Stohr, Alexander Frömmgen, Amr Rizk, Michael Zink, Ralf Steinmetz, and Wolfgang Effelsberg. "Where Are the Sweet Spots? A Systematic Approach to Reproducible DASH Player Comparisons." In: *Proceedings of ACM Multimedia*. 30749. ACM, 10/2017 (cit. on pp. 62, 69, 71).

[Sto+17b] Denny Stohr, Iva Toteva, Stefan Wilk, Wolfgang Effelsberg, and Ralf Steinmetz. "User-Generated Video Composition Based on Device Context Measurements." In: *International Journal of Semantic Computing* Volume 11 (Issue 01 2017). C3, pp. 1–19 (cit. on pp. 28, 37).

[Sto+18] Denny Stohr, Timo Kalle, Andreas U. Mauthe, Amr Rizk, Ralf Steinmetz, and Wolfgang Effelsberg. "Towards Improved DASH Adaptation in NDN: An Emulative Analysis." In: *IEEE Local Computer Networks*. C3. Chicago, USA, 10/2018 (cit. on p. 87).

[SUS16] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. "BOLA: Near-Optimal Bitrate Adaptation for Online Videos." In: *Proceedings of IEEE INFOCOM 2016*. IEEE, 04/2016, pp. 1–9 (cit. on pp. 30 sq., 65, 89).

[SWE14] Denny Stohr, Stefan Wilk, and Wolfgang Effelsberg. "Monitoring of User Generated Video Broadcasting Services." In: *Proceedings of the First International Workshop on Internet-Scale Multimedia Management*. WISMM '14. New York, NY, USA: ACM, 2014, pp. 39–42 (cit. on p. 37).

[SWH17] Matthias Schulz, Daniel Wegemer, and Matthias Hollick. *Nexmon: The C-Based Firmware Patching Framework*. 2017. URL: https://nexmon.org (cit. on p. 9).

[Tut99] Walter HW Tuttlebee. "Software-Defined Radio: Facets of a Developing Technology." In: *IEEE Personal Communications* 6.2 (1999). 00254, pp. 38–44 (cit. on p. 9).

[Ulv10] T. Ulversoy. "Software Defined Radio: Challenges and Opportunities." In: *IEEE Communications Surveys Tutorials* 12.4 (Fourth 2010), pp. 531–550 (cit. on p. 9).

[Wan+04] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity." In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612 (cit. on p. 21).

[WE14a] Stefan Wilk and Wolfgang Effelsberg. "Mobile Video Broadcasting Services - Combining Video Composition and Network Efficient Transmission." In: *Proceedings of the ACM International Conference on Multimedia*. 2014 (cit. on pp. 28 sq.).

[WE14b] Stefan Wilk and Wolfgang Effelsberg. "Systematic Assessment of the Video Recording Position for User-Generated Event Videos." In: *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 1009–1012 (cit. on p. 49).

[WE14c] Stefan Wilk and Wolfgang Effelsberg. "The Influence of Camera Shakes, Harmful Occlusions and Camera Misalignment on the Perceived Quality in User Generated Video." In: *2014 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 07/2014, pp. 1–6 (cit. on pp. 28 sq.).

[Wil16] Stefan Wilk. "Quality-Aware Content Adaptation in Digital Video Streaming." TU Darmstadt, 2016 (cit. on pp. 18, 32, 37, 105).

[WK01] Andrew B. Watson and Lindsay Kreslake. "Measurement of Visual Impairment Scales for Digital Video." In: *Proceedings of the SPIE: Human Vision and Electronic Imaging VI*. International Society for Optics and Photonics, 2001, pp. 79–89 (cit. on p. 58).

[WKE15] Stefan Wilk, Stephan Kopf, and Wolfgang Effelsberg. "Video Composition in the Crowd: A System to Compose User-Generated Videos in Near Real-Time." In: *Proceedings of the ACM International Conference on Multimedia Systems*. 2015 (cit. on p. 51).

[WM08] Stefan Winkler and Praveen Mohandas. "The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics." In: *IEEE Transactions on Broadcasting* 54.3 (09/2008), pp. 660–668 (cit. on p. 22).

[WMW17] S. Waldmann, K. Miller, and A. Wolisz. "Traffic Model for HTTP-Based Adaptive Streaming." In: *2017 IEEE Conference on Computer Communications Workshops (INFOCOM Workshops)*. 05/2017, pp. 683–688 (cit. on p. 32).

[WRZ16] Cong Wang, Amr Rizk, and Michael Zink. "SQUAD: A Spectrum-Based Quality Adaptation for Dynamic Adaptive Streaming over HTTP." In: *Proceedings of ACM MMSys*. 2016, 1:1–1:12 (cit. on pp. 20, 30 sq., 67, 70, 95).

[WSE16] Stefan Wilk, Denny Stohr, and Wolfgang Effelsberg. "A Content-Aware Video Adaptation Service to Support Mobile Video." In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 12 (5s 11/2016), 82:1–82:23 (cit. on pp. 20, 30).

[WZE16] Stefan Wilk, Roger Zimmermann, and Wolfgang Effelsberg. "Leveraging Transitions for the Upload of User-Generated Mobile Video." In: *Proceedings of the 8th International Workshop on Mobile Video*. MoVid '16. ACM, 2016, 5:1–5:6 (cit. on p. 29).

[XWP03] Jing Xu, Murray Woodside, and Dorina Petriu. "Performance Analysis of a Software Design Using the UML Profile for Schedulability, Performance and Time." In: *Computer Performance Evaluation. Modelling Techniques and Tools*. Ed. by Peter Kemper and William H. Sanders. Vol. 2794. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 291–307 (cit. on p. 19).

[Xyl+14] George Xylomenos, Christopher N. Ververidis, Vasilios A. Siris, Nikos Fotiou, et al. "A Survey of Information-Centric Networking Research." In: *IEEE Communications Surveys & Tutorials* 16.2 (2014), pp. 1024–1049 (cit. on p. 9).

[Yin+15] X Yin, A Jindal, V Sekar, and B Sinopoli. "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP." In: *Proceedings of ACM SIGCOMM*. 00165. 2015, pp. 325–338 (cit. on p. 30).

[Zab+17] Anatoliy Zabrovskiy, Evgeny Kuzmin, Evgeny Petrov, Christian Timmerer, and Christopher Mueller. "AdViSE." In: *Proceedings of ACM MMSys*. New York, New York, USA: ACM Press, 2017, pp. 217–220 (cit. on p. 33).

[Zha+17] Haibo Zhang, Prasanna Venkatesh Rengasamy, Shulin Zhao, Nachiappan Chidambaram Nachiappan, et al. "Race-to-Sleep + Content Caching + Display Caching: A Recipe for Energy-Efficient Video Streaming on Handhelds." In: *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-50 '17. New York, NY, USA: ACM, 2017, pp. 517–531 (cit. on p. 26).

[Zhe+16] Kan Zheng, Zhe Yang, Kuan Zhang, Periklis Chatzimisios, Kan Yang, and Wei Xiang. "Big Data-Driven Optimization for Mobile Networks toward 5G." In: *IEEE Network* 30.1 (01/2016), pp. 44–51 (cit. on p. 23).

[Zin+03] Michael Zink, Oliver Künzel, Jens Schmitt, and Ralf Steinmetz. "Subjective Impression of Variations in Layer Encoded Videos." In:

IWQoS. Ed. by Kevin Jeffay, Ion Stoica, and Klaus Wehrle. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 137–154 (cit. on p. 20).

[ZL15] Cong Zhang and Jiangchuan Liu. "On Crowdsourced Interactive Live Streaming: A Twitch.Tv-Based Measurement Study." In: *Proceedings of the ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. NOSSDAV '15. ACM, 2015, pp. 55–60 (cit. on pp. 26, 40, 43).

[ZSS05] Michael Zink, Jens Schmitt, and Ralf Steinmetz. "Layer-Encoded Video in Scalable Adaptive Streaming." In: *IEEE Transactions on Multimedia* 7.1 (02/2005), pp. 75–83 (cit. on p. 78).

Author's Publications

Main Publications

[Sto+15a] Denny Stohr, Tao Li, Stefan Wilk, Silvia Santini, and Wolfgang Effelsberg. "An Analysis of the YouNow Live Streaming Platform." In: *Proceedings of the 9th IEEE Workshop on Network Measurements (WNM)*. B1, C03. Clearwater, USA, 10/2015 (cit. on p. 37).

[Sto+15b] Denny Stohr, Matthias Schulz, Matthias Hollick, and Wolfgang Effelsberg. "APP and PHY in Harmony: Demonstrating Scalable Video Streaming Supported by Flexible Physical Layer Control." In: *Proceedings of IEEE WoWMoM 2015 (Demos)*. The first two authors contributed equally. Boston, USA: IEEE, 06/2015, pp. 1–3 (cit. on p. 101).

[Sto+16a] Denny Stohr, Fares Beji, Rahul Dwarakanath, Ralf Steinmetz, and Wolfgang Effelsberg. "Mobile NDN-Based Dynamic Adaptive Streaming over HTTP." In: *Proceedings of the IEEE Conference on Local Computer Networks (LCN)*. (Demo) C03. Duabi: IEEE, 2016, p. 3 (cit. on pp. 87 sq.).

[Sto+16b] Denny Stohr, Alexander Frömmgen, Jan Fornoff, Michael Zink, Alejandro Buchmann, and Wolfgang Effelsberg. "QoE Analysis of DASH Cross-Layer Dependencies by Extensive Network Emulation." In: *Proceedings of the SIGCOMM Internet-QoE Workshop*. Florianopolis: ACM, 2016 (cit. on pp. 62, 69).

[Sto+16c] Denny Stohr, Stefan Wilk, Iva Toteva, Wolfgang Effelsberg, and Ralf Steinmetz. "Context Not Content: A Novel Approach to Real-Time User-Generated Video Composition." In: *Proceedings of the 2016 IEEE International Symposium on Multimedia*. San Jose, CA, USA: IEEE, 2016, p. 6 (cit. on pp. 28, 37).

[Sto+17a] Denny Stohr, Alexander Frömmgen, Amr Rizk, Michael Zink, Ralf Steinmetz, and Wolfgang Effelsberg. "Where Are the Sweet Spots? A Systematic Approach to Reproducible DASH Player Comparisons." In: *Proceedings of ACM Multimedia*. 30749. ACM, 10/2017 (cit. on pp. 62, 69, 71).

[Sto+17b] Denny Stohr, Iva Toteva, Stefan Wilk, Wolfgang Effelsberg, and Ralf Steinmetz. "User-Generated Video Composition Based on Device Context Measurements." In: *International Journal of Semantic*

Computing Volume 11 (Issue 01 2017). C3, pp. 1–19 (cit. on pp. 28, 37).

[Sto+18] Denny Stohr, Timo Kalle, Andreas U. Mauthe, Amr Rizk, Ralf Steinmetz, and Wolfgang Effelsberg. "Towards Improved DASH Adaptation in NDN: An Emulative Analysis." In: *IEEE Local Computer Networks*. C3. Chicago, USA, 10/2018 (cit. on p. 87).

[SWE14] Denny Stohr, Stefan Wilk, and Wolfgang Effelsberg. "Monitoring of User Generated Video Broadcasting Services." In: *Proceedings of the First International Workshop on Internet-Scale Multimedia Management*. WISMM '14. New York, NY, USA: ACM, 2014, pp. 39–42 (cit. on p. 37).

Co-authored Publications

[Frö+16] Alexander Frömmgen, Denny Stohr, Jan Fornoff, Wolfgang Effelsberg, and Alejandro Buchmann. "Capture and Replay: Reproducible Network Experiments in Mininet." In: *Proceedings of the ACM SIGCOMM Conference (Demo)*. 2016, pp. 621–622.

[Frö+18] Alexander Frömmgen, Denny Stohr, Boris Koldehofe, and Amr Rizk. "Don't Repeat Yourself: Seamless Execution and Analysis of Extensive Network Experiments." In: 2018. arXiv: 1802.03455 (cit. on pp. 62 sq.).

[Koc+18] Christian Koch, Moritz Lode, Denny Stohr, Amr Rizk, and Ralf Steinmetz. "Collaborations on YouTube: From Unsupervised Detection to the Impact on Video and Channel Popularity." In: 05/01/2018. arXiv: 1805.01887 [cs].

[Ric+14] Björn Richerzhagen, Stefan Wilk, Julius Rückert, Wolfgang Effelsberg, and Denny Stohr. "Transitions in Live Video Streaming Services Categories and Subject Descriptors." In: *ACM VideoNext 2014*. ACM, 12/2014, pp. 37–38 (cit. on p. xxxiv).

[Sch+15] Matthias Schulz, Denny Stohr, Stefan Wilk, Benedikt Rudolph, Wolfgang Effelsberg, and Matthias Hollick. "APP and PHY in Harmony: A Framework Enabling Flexible Physical Layer Processing to Address Application Requirements," in: *Proceedings of the International Conference on Networked Systems (NetSys)*. 03/2015 (cit. on p. xxxiii).

[Wil+15a] Stefan Wilk, Julius Rückert, Denny Stohr, Björn Richerzhagen, and Wolfgang Effelsberg. "Efficient Video Streaming through Seamless Transitions Between Unicast and Broadcast (Demo)." In: *Proceedings of the International Conference on Networked Systems (NetSys)*. Cottbus, Deutsachland, 03/2015 (cit. on p. xxxiii).

[Wil+15b] Stefan Wilk, Sophie Schönherr, Denny Stohr, and Wolfgang Effelsberg. "EnvDASH - An Environment-Aware Dynamic Adaptive Streaming over HTTP System." In: *Procedings of TVX*. 2015, pp. 113–118.

[Wil+16] Stefan Wilk, Dominik Schreiber, Denny Stohr, and Wolfgang Effelsberg. "On the Effectiveness of Video Prefetching Relying on Recommender Systems for Mobile Devices." In: *Proceedings of IEEE CCNC*. IEEE, 01/2016, pp. 429–434.

[WSE15] Stefan Wilk, Denny Stohr, and Wolfgang Effelsberg. "VAS: A Video Adaptation Service to Support Mobile Video." In: *Proceedings of the 25th Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*. 03/2015.

[WSE16] Stefan Wilk, Denny Stohr, and Wolfgang Effelsberg. "A Content-Aware Video Adaptation Service to Support Mobile Video." In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 12 (5s 11/2016), 82:1–82:23 (cit. on pp. 20, 30).

List of Acronyms

AA	Adaptation Algorithm
ASIC	Application Specific Integrated Circuit
AVC	Advanced Video Coding
AVS	Adaptive Video Streaming
BBA	Buffer-based Adaptation
BBR	Bottleneck Bandwidth And Round-trip Propagation Time
CC	Congestion Control
CCN	Content-Centric Networking
CDN	Content Delivery Network
CS	Content Store
cwnd	Congestion Window Size
DASH	Dynamic Adaptive Streaming Over HTTP
ECDF	Empirical Cumulative Density Function
EWMA	Exponentially Weighted Moving Average
FEC	Forward Error Correction
FIB	Forwarding Information Base
FPGA	Field Programmable Gate Array
FPS	Frames Per Second
HAS	HTTP Adaptive Streaming
HLS	Apple HTTP Live Streaming
HTTP	Hypertext Transfer Protocol
HVS	Human Visual System
ICN	Information-centric Network
ITU	International Telecommunication Union
JND	Just Noticeable Difference
LiViU	Live Video Upload System
MBS	Mobile Video Broadcasting Service
MOS	Mean Opinion Score
MPD	Media Presentation Description
MPTCP	Multipath-TCP
MSR	Mean Squared Error

MVC	Mobile Video Composition
NDN	Named Data Networking
OTT	Over-The-Top
PIT	Pending Interest Table
PoI	Point Of Interest
QoE	Quality Of Experience
QoS	Quality Of Service
QUIC	Quick UDP Internet Connections
REST	Representational State Transfer
RTCP	Real-Time Control Protocol
RTMP	Real-Time Messaging Protocol
RTP	Real-Time Transport Protocol
RTSP	Real-Time Streaming Protocol
RTT	Round Trip Time
SDR	Software-Defined Radio
SQUAD	Spectrum-based Quality Adaptation For DASH
SSCQS	Single Stimulus Continuous Quality Scale
SVC	Scalable Video Coding
TBA	Throughput-based Adaptation
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UGV	User-Generated Video
VoD	Video On Demand
VRS	Video Representation Switches
VSS	Video Streaming System

Supervised Student Theses

[Bäh16] Timo Bähr. "Aggregating high-level context information from heterogeneous sensor sources by a novel mobile sensing framework." Supervisor: Stefan Wilk, Co-Supervisor: Denny Stohr. Master Thesis. TU Darmstadt, 01/01/2016.

[For16] Jan Fornoff. "Retrieving Adaptation Knowledge by Offline Simulation in MPEG DASH Video Streaming." Supervisor: Alexander Frömmgen, Co-Supervisor: Denny Stohr. TU Darmstadt, 01/11/2016.

[Kal17] Timo Kalle. "DASH Adaption Algorithms in Named Data Networking." Supervisor: Denny Stohr, Co-Supervisor: Amr Rizk. Bachelor Thesis. TU Darmstadt, 2017.

[Rei17] Julian Reichwein. "Einflussfaktoren von Pitch-Videos Auf Den Erfolg von Crowdfunding- Kampagnen." Supervisor: Michael Wessel, Co-Supervisor: Denny Stohr. Bachelor Thesis. TU Darmstadt, 2017.

[Sch15] Dominik Schreiber. "Recommendation-Aware Mobile Prefetching Strategies for Video Sharing Sites." Supervisor: Denny Stohr, Co-Supervisor: Stefan Wilk. Master Thesis. TU Darmstadt, 2015. 74 pp.

[Sch17] Simon Schindel. "Cache-Aided DASH in 5G Networks Using HTTP/2 Server Push." Supervisor: Christian Koch, Co-Supervisor: Denny Stohr. Master Thesis. TU Darmstadt, 2017.

[Tra17] Mai Ly Tran. "Retrieving DASH Session QoE Metrics from Real-World Measurements Using Machine Learning Approaches." Supervisor: Denny Stohr. Bachelor Thesis. TU Darmstadt, 2017.

Curriculum Vitæ

Personal Information

Name	Denny Stohr
Date of Birth	July 19, 1988
Place of Birth	Mainz
Nationality	German

Education

01/2014 – 06/2018	Technische Universität Darmstadt Doctoral candidate – Department of Computer Science
09/2011 – 12/2013	Universität Mannheim Information Systems – Degree: Masters of Science
09/2008 – 08/2011	Universität Mannheim Information Systems – Degree: Bachelor of Science

Professional Experience

01/2014 – 06/2018	Technische Universität Darmstadt Research assistant – Distributed Multimedia Systems (DMS), Multimedia Communications Lab (KOM)

Awards and Honors

2015	Best Paper Award: Matthias Schulz, Denny Stohr, et al. "APP and PHY in Harmony: A Framework Enabling Flexible Physical Layer Processing to Address Application Requirements," in: *Proceedings of the International Conference on Networked Systems (NetSys)*. 03/2015
2015	Best Demo Award (3rd Place): Stefan Wilk, Julius Rückert, et al. "Efficient Video Streaming through Seamless Transitions Between Unicast and Broadcast (Demo)." In: *Proceedings of the International Conference on Networked Systems (NetSys)*. Cottbus, Deutsachland, 03/2015

2014	Best Demo Award: Björn Richerzhagen, Stefan Wilk, et al. "Transitions in Live Video Streaming Services Categories and Subject Descriptors." In: *ACM VideoNext 2014*. ACM, 12/2014, pp. 37–38

Teaching Activities

Since 2013	Technische Universität Darmstadt Tutor for various Bachelor-, Master Theses
Since 2014	Technische Universität Darmstadt Organizer for the Seminar Distributed Multimedia Systems

Erklärung laut §9 der Promotionsordnung

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 2018

Denny Stohr

Colophon

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography ''*The Elements of Typographic Style*''.

We attribute the Icons and resources used in Figures as follows (if not specified otherwise, all Icons have been obtained from https://www.flaticon.com):

Icons used in Figure 1 are created by Picol and Vectors Market licensed under `CC 3.0 BY` and by Freepik under basic licensing.

Icons used in Figures 4, 33 and 54 are obtained from https://aws.amazon.com/architecture/icons/

Icons used in Figure 7 are created by Picol licensed under `CC 3.0 BY` and by Freepik under basic licensing.

Icons used in Figure 10 are created by Picol and AnhGreen licensed under `CC 3.0 BY` and by Freepik under basic licensing.

Icons used in Figure 22 are created AnhGreen licensed under `CC 3.0 BY` and by Freepik under basic licensing.

Icons used in Figures 24 and 54 are created by Picol licensed under `CC 3.0 BY` and by Freepik under basic licensing.

Figure 25 has been created on http://maps.stamen.com/

Icons used in Figure 35 are made by Smashicons and Freepik under Freepik basic licensing.

Data analysis and plotting for this thesis has been conducted using `matplotlib`, `searborn`, `pandas`, `scipy`, `numpy`, `scikit-learn` and `statsmodels` Python libraries mostly within an Jupyter Notebook environment. I want to express my sincere gratitude to all developers and maintainers for providing these great tools.

For further resources related to this thesis, please visit https://www.dstohr.de/PhD.